Doc No: UG60917-IDP-030
Rev A
Date: 08 February 2012
Contract No. NNG10XA03C

JOINT POLAR SATELLITE SYSTEM (JPSS)
COMMON GROUND SYSTEM (CGS)

IDPS ING SOFTWARE USER'S MANUAL
PART 2

USER GUIDE

CDRL No. A032

RAYTHEON COMPANY
INTELLIGENCE AND INFORMATION SYSTEMS (IIS)
JPSS CGS PROGRAM
AURORA, COLORADO

Copyright © 2010-2012
Raytheon Company
Unpublished Work
ALL RIGHTS RESERVED

This data was developed pursuant to Contract Number NNG10XA03C with the US Government. The US government's rights in and to this copyrighted data are as specified in DFAR 252.227-7013 which was made part of the above contract.

IAW DFARS 252.227-7036, Raytheon hereby declares that, to the best of its knowledge and belief, the technical data delivered under Contract No. NNG10XA03C is complete, accurate, and complies with all requirements of the Contract.

TITLE: JOINT POLAR SATELLITE SYSTEM (JPSS) COMMON GROUND SYSTEM (CGS) IDPS ING SW USER MANUAL PART 2

APPROVAL SIGNATURES:
(Signatures indicate compliance with these requirements)

William Sullivan

Digitally signed by William Sullivan
DN: cn=William Sullivan, o=JPSS, ou=DCMS, email=wjsullivan@raytheon.com, c=US
Date: 2012.02.17 16:10:37 -07'00'

William J. Sullivan Date
Program Manager

Frank Zorn

Digitally signed by Frank Zorn
DN: cn=Frank Zorn, o=JPSS Sustainment, ou=JPSS Sustainment, email=fgzorn@raytheon.com, c=US
Date: 2012.02.14 09:06:16 -07'00'

Frank Zorn Date
Sustainment IPT Lead

Harvey Lee

Digitally signed by Harvey Lee
DN: cn=Harvey Lee, o=JPSS, ou=JPSS, email=hylee@raytheon.com, c=US
Date: 2012.02.15 10:26:05 -07'00'

Harvey Lee Date
IDPS Sustainment Lead

Justin Lee

Digitally signed by Justin Lee
DN: cn=Justin Lee, o=IDPS Sustainment, ou=JPSS, email=justin.lee@raytheon.com, c=US
Date: 2012.02.13 07:45:36 -07'00'

Justin Lee Date
Sustainment Software Manager

Sheryl Fertman

Digitally signed by Sheryl Fertman
DN: cn=Sheryl Fertman, o=IIS QA, ou=IIS QA, email=slfertman@raytheon.com, c=US
Date: 2012.02.15 10:37:29 -07'00'

Sheryl L Fertman Date
IIS Quality

Revision/Change History

Revision	Document Date	Revision/Change Description	Pages Affected
-	18 Nov 2010	This document was numbered under the NPOESS contract as UG60822-IDP-030 Rev A. Initial release per ECR-IDP-R1303.	All
A	08 Feb 2012	Updates per ECR-IDP-004303 for Maintenance Release 6 (PCR25383, PCR25908, PCR27242, PCR27709, General Cleanup and corrections)	Various

Table of Contents

1.0 SCOPE 1
1.1 IDENTIFICATION 1
1.2 SYSTEM OVERVIEW 1
1.3 DOCUMENT OVERVIEW 4
2.0 REFERENCE DOCUMENTS 5
3.0 SOFTWARE SUMMARY 6
3.1 SOFTWARE APPLICATION 6
3.1.1 SMD 6
3.1.2 MSD 6
3.2 SOFTWARE ORGANIZATION AND OVERVIEW OF OPERATION 7
3.2.1 Logical Components of the Software / Process Architecture 7
3.2.1.1 SMD 7
3.2.1.2 MSD 7
3.2.2 Performance Characteristics 8
3.2.2.1 SMD 8
3.2.2.1.1 C3S Socket Interface 8
3.2.2.1.1.1 Socket Buffer Size Tuning 8
3.2.2.1.1.2 Round Robin Socket Approach 8
3.2.2.1.1.3 Missing sockets 8
3.2.2.1.1.4 Keep Alive settings 9
3.2.2.1.2 GCOM File Interface 9
3.2.2.1.3 Asynchronous Writes 10
3.2.2.1.4 Shell RDRs 10
3.2.2.2 MSD 10
3.2.2.2.1 Software Update as offline process 11
3.2.3 Interfaces to Other Systems 11
3.2.4 Control 14
3.3 ALTERNATE MODES OF OPERATION 15
3.3.1 Lightweight mode 15
3.3.2 SMD 15

3.3.2.1 Retransmits 15
3.3.2.2 Extended Headers 15
3.3.3 MSD 16
3.3.3.1 Persistent Mode 16
3.3.3.2 Transient Mode 16
3.4 SECURITY 16
4.0 BASIC CONCEPTS 17
4.1 STARTUP AND SHUTDOWN 17
4.1.1 Startup 17
4.1.1.1 SMD 17
4.1.1.1.1 Processes 17
4.1.1.1.2 Modes 18
4.1.1.2 MSD 18
4.1.1.2.1 Processes 18
4.1.1.2.2 Modes 18
4.1.2 SHUTDOWN 18
4.2 USER INTERFACE OVERVIEW 19
4.3 RELATED PROCESSING 19
4.4 DATA BACKUP 19
5.0 REFERENCE GUIDE 20
5.1 SMD 20
5.1.1 Startup parameters 20
5.1.2 Application Packets 21
5.1.3 Data Provider 21
5.1.3.1 C3S interface 21
5.1.3.1.1 Extended Application Packets 21
5.1.3.1.2 Socket Connections 21
5.1.3.1.2.1 Domain Naming Service 22
5.1.3.1.3 Retransmit Requests 23
5.1.3.1.3.1 Port State Files 24
5.1.3.2 GCOM 24
5.1.3.2.1 File-based 24

5.1.3.2.2 Landing Zone 24
5.1.4 Drop Packets 25
5.1.5 Packet Timestamps for Segmented Packets 25
5.1.6 Timestamp Delta Modification 26
5.1.7 RDRs 26
5.1.7.1 Common Concepts 26
5.1.7.1.1 Sizing and structure 26
5.1.7.1.2 Granule IDs 27
5.1.7.1.3 RDR Timeouts and Thresholds 27
5.1.7.1.3.1 Primary Timeout/Threshold 28
5.1.7.1.3.2 Repair Timeout/Threshold 28
5.1.7.1.3.3 Unlock Timeout 28
5.1.7.1.3.4 RDR Full Timeout 29
5.1.7.1.4 RDR Versioning 29
5.1.7.1.5 RDR Completion Calculations 29
5.1.7.1.5.1 Day/Night 30
5.1.7.1.5.2 Short Granules 30
5.1.7.1.5.3 Fattened Granules 30
5.1.7.1.5.4 Packets with Fill 31
5.1.7.1.5.5 Asynchronous Packets 31
5.1.7.1.5.6 Super-Segmented Packets 31
5.1.7.2 Specific RDRs 31
5.1.7.2.1 ATMS Science 32
5.1.7.2.1.1 Fattened Granule 32
5.1.7.2.2 ATMS Diagnostic 32
5.1.7.2.2.1 Short Granule 32
5.1.7.2.3 ATMS Dwell 32
5.1.7.2.4 ATMS Memory Dump 32
5.1.7.2.4.1 Asynchronous Packet 32
5.1.7.2.5 ATMS Telemetry 33
5.1.7.2.6 CERES Science 33
5.1.7.2.6.1 Short Granule 33

5.1.7.2.7 CERES Diagnostic 33
5.1.7.2.7.1 Short Granule 33
5.1.7.2.8 CERES Telemetry 33
5.1.7.2.8.1 Short Granule 33
5.1.7.2.9 CrIS Science 34
5.1.7.2.9.1 Fattened Granule 34
5.1.7.2.9.2 Asynchronous Packet 34
5.1.7.2.10 CrIS Diagnostic 34
5.1.7.2.10.1 Short Granule 34
5.1.7.2.11 CrIS HK Dwell 34
5.1.7.2.12 CrIS IM Dwell 34
5.1.7.2.13 CrIS SSM Dwell 35
5.1.7.2.14 CrIS Memory Dump 35
5.1.7.2.14.1 Aysnchronous Packet 35
5.1.7.2.15 CrIS Telemetry 35
5.1.7.2.15.1 Fattened Granule 35
5.1.7.2.15.2 Short Granule 35
5.1.7.2.16 OMPS FSW Bootup Status 35
5.1.7.2.16.1 Aysnchronous Packet 36
5.1.7.2.17 OMPS Dwell 36
5.1.7.2.18 OMPS Memory Dump 36
5.1.7.2.18.1 Asynchronous Packet 36
5.1.7.2.18.2 Super-Segmented Packet 36
5.1.7.2.19 OMPS Telemetry 36
5.1.7.2.20 OMPS LP Science 36
5.1.7.2.20.1 Short Granule 37
5.1.7.2.20.2 Super-Segmented Packet 37
5.1.7.2.21 OMPS LP Calibration 37
5.1.7.2.21.1 Super-Segmented Packet 37
5.1.7.2.22 OMPS LP Diagnostic Exposure One 37
5.1.7.2.22.1 Super-Segmented Packet 38
5.1.7.2.23 OMPS LP Diagnostic Exposure Two 38

5.1.7.2.23.1 Super-Segmented Packet 38
5.1.7.2.24 OMPS LP Diagnostic Calibration 38
5.1.7.2.24.1 Super-Segmented Packet 38
5.1.7.2.25 OMPS NP Science 39
5.1.7.2.25.1 Short Granule 39
5.1.7.2.25.2 Super-Segmented Packet 39
5.1.7.2.26 OMPS NP Calibration 39
5.1.7.2.26.1 Super-Segmented Packet 39
5.1.7.2.27 OMPS NP Diagnostic Earth View 40
5.1.7.2.27.1 Super-Segmented Packet 40
5.1.7.2.28 OMPS NP Diagnostic Calibration 40
5.1.7.2.28.1 Super-Segmented Packet 40
5.1.7.2.29 OMPS TC Science 40
5.1.7.2.29.1 Short Granule 41
5.1.7.2.29.2 Super-Segmented Packet 41
5.1.7.2.30 OMPS TC Calibration 41
5.1.7.2.30.1 Super-Segmented Packet 41
5.1.7.2.31 OMPS TC Diagnostic Earth View 41
5.1.7.2.31.1 Super-Segmented Packet 42
5.1.7.2.32 OMPS TC Diagnostic Calibration 42
5.1.7.2.32.1 Super-Segmented Packet 42
5.1.7.2.33 Spacecraft Diary 42
5.1.7.2.33.1 Fattened Granule 42
5.1.7.2.34 Spacecraft Telemetry 43
5.1.7.2.34.1 Fattened Granule 43
5.1.7.2.34.2 Asynchronous Packets 43
5.1.7.2.35 VIIRS Science 43
5.1.7.2.35.1 Short Granule 43
5.1.7.2.35.2 Day/Night 43
5.1.7.2.35.3 Asynchronous packets 44
5.1.7.2.36 VIIRS Diagnostic 44
5.1.7.2.36.1 Short Granule 44

5.1.7.2.36.2 Asynchronous packets 44
5.1.7.2.37 VIIRS Memory Dump 44
5.1.7.2.37.1 Asynchronous Packet 44
5.1.7.2.38 VIIRS Telemetry 45
5.1.7.2.38.1 Short Granule 45
5.1.7.2.39 VIIRS Diagnostic Telemetry 45
5.1.7.2.39.1 Short Granule 45
5.1.7.2.40 AMSR2 Science 45
5.1.7.2.40.1 Fattened Granule 45
5.1.7.2.41 AMSR2 Telemetry 45
5.1.7.2.41.1 Fattened Granule 46
5.2 MSD 47
5.2.1 Startup parameters 47
5.2.2 File Naming conventions 47
5.2.3 Landing Zones 47
5.2.3.1 Persistent Mode 48
5.2.3.2 Transient Mode 48
5.2.4 File Completion Strategies 49
5.2.4.1 Suffix Completion 49
5.2.4.2 Checksum Completion 49
5.2.5 Polling of landing zones 49
5.2.6 Purge of older versions 50
5.2.7 Preventing older versions 50
5.2.8 Location of erroneous MSD 51
5.2.9 Effectivity 51
5.2.10 Forecast data 51
5.2.10.1 Effectivity 52
5.2.10.2 Minimum time until use 52
5.2.10.3 Extended forecasts 52
5.2.10.4 Forecast Based Collection Short Name variations 53
5.2.10.5 Substitute Data Based Collection Short Names 53
5.2.10.6 Temporal Interpolation 53

5.2.11 Configuring landing zone monitoring 54
6.0 MESSAGES AND ERRORS 57
6.1 ERROR RECOVERY 57
6.1.1 SMD 57
6.1.2 MSD 57
6.2 MESSAGES 59
6.2.1 Status/Details 59
6.2.2 Performance 59
6.2.3 Debug 59
6.3 STATUS/DETAILS MESSAGE CONTENT 60
6.3.1 Common 60
6.3.1.1 PROC_INIT_COMPL 60
6.3.1.2 ING_ERR 60
6.3.1.3 ING_INFO 61
6.3.2 SMD Specific 62
6.3.2.1 ING_FREE_FORM 62
6.3.2.2 DROP_PKT 63
6.3.2.3 RDR_START 64
6.3.2.4 RDR_FAIL 65
6.3.2.5 RDR_GRAN 66
6.3.2.6 RDR_PARTIAL 67
6.3.3 MSD Specific 68
6.3.3.1 ANC_RECV 68
6.3.3.2 ANC_FAIL 68
6.3.3.3 MSNSCHED_RECV 69
6.3.3.4 MSNSCHED_FAIL 69
6.3.3.5 SCDB_RECV 70
6.3.3.6 SCDB_FAIL 70
6.3.3.7 MSD_RECV 71
6.3.3.8 MSD_FAIL 71
6.3.3.9 STATIC_RECV 72
6.3.3.10 STATIC_FAIL 72

6.3.3.11 TLE_RECV 73
6.3.3.12 TLE_FAIL 73
7.0 APPENDIX A - SOFTWARE MAINTENANCE 74
7.1 UPDATING PROCESSING COEFFICIENTS 74
7.1.1 Changes to values only 74
7.1.2 Adding/Deleting parameters 74
7.1.2.1 Updating the structure file 74
7.1.2.2 Updating the converter 75
7.1.2.2.1 Stand-alone values 75
7.1.2.2.2 Arrays 75
7.1.2.2.2.1 One dimensional 75
7.1.2.2.2.2 Two dimensional 76
7.1.2.2.2.3 Three Dimensional 77
7.1.3 New Processing Coefficient file 77
7.1.4 Testing 77
7.1.4.1 Updating the unit test 77
7.1.4.2 Executing the unit test 79
7.1.5 Providing values to DQM 79

List of Figures

FIGURE 1.2-1 ING SUBSYSTEM IN CONTEXT 1
FIGURE 3.2.1-1 ING PROCESS ARCHITECTURE 7
FIGURE 3.2.3-1 INF TO ING INTERFACES 12
FIGURE 3.2.3-2 DMS/DQM/DDS TO ING INTERFACES 13
FIGURE 3.2.3-3 C3S TO ING INTERFACE 14
FIGURE 3.2.3-4 SOFTWARE UPDATE INTERFACE 14

List of Tables

TABLE 2.0-1 REFERENCED DOCUMENTS 5
TABLE 4.1.1.1.1-1 SMD PROCESS CONFIGURATIONS 17
TABLE 4.1.1.2.1-1 MSD PROCESS CONFIGURATIONS 18
TABLE 5.1.1-1 SMD STARTUP PARAMETERS 20

1.0 SCOPE

1.1 IDENTIFICATION

The Interface Data Processing Segment (IDPS) Ingest (ING) Software User's Manual Part I describes the architecture, installation and setup of the Ingest Subsystem. This document has been prepared in support of the ING Software Item (SI), of the Joint Polar Satellite System (JPSS).

CDRL No. A032 refers to document number UG60917-IDP-030, titled JPSS Common Ground System (CGS) IDPS ING Software User's Manual Part 2.

1.2 SYSTEM OVERVIEW

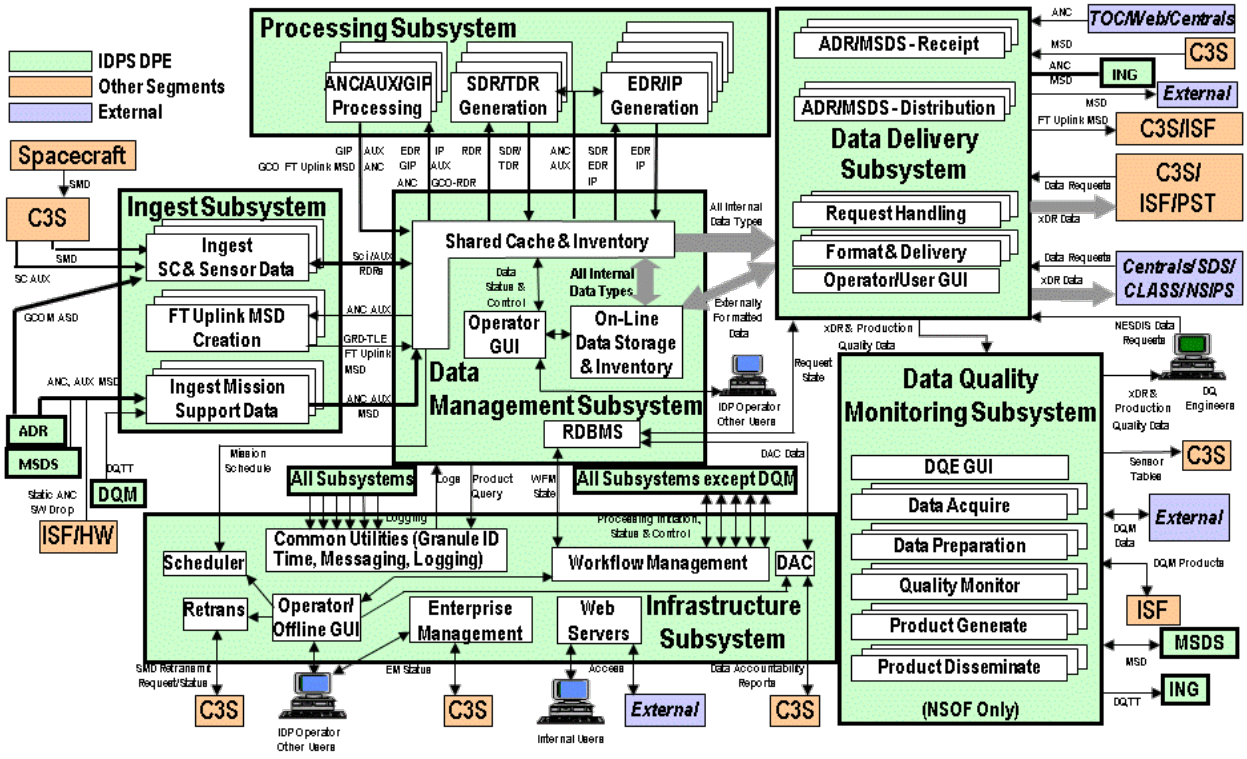

FIGURE 1.2-1 ING SUBSYSTEM IN CONTEXT

The Joint Polar Satellite System (JPSS) is a System-of-Systems architecture whose mission is to provide military and civilian agencies with environmental, meteorological, and climatological data and products. The Common Ground System (CGS) provides command and control of the National Polar-orbiting Operational Environmental Satellite System (NPOESS) Preparatory Project (NPP) and JPSS spacecraft and instruments. The CGS transports commands, telemetry, and mission data between the space, ground control, and processing facilities and

the Customer locations, and enables mission planning, asset management, and generation and delivery of the mission products.

The CGS Program is responsible for the design, implementation, integration, test, deployment, and transition of the three JPSS Ground Segments: the Command, Control, and Communications Segment (C3S), the Interface Data Processing Segment (IDPS), and the Field Terminal Segment (FTS). C3S manages the overall JPSS CGS mission, including active management and accounting of mission data, from storage on the satellite solid-state recorder to delivery to the IDPS. The IDPS is responsible for receiving raw mission data from C3S and for creating and delivering useable environmental products to Users. FTS provides hardware and software specifications needed to build JPSS CGS-compliant field terminals (FTs), along with updated processing software designed to handle direct broadcast data streams.

The IDPS of JPSS CGS is responsible for providing data records to four meteorological centrals: Air Force Weather Agency (AFWA); National Environmental Satellite, Data, and Information Service (NESDIS); Fleet Numerical Meteorology and Oceanography Center (FNMOC); and Naval Oceanographic Office (NAVOCEANO)) on a time critical basis. The IDPS ingests and processes Stored Mission Data (SMD) received from the Command, Control and Communications Segment (C3S). The artifacts from satellite onboard storage and ground communication routing are removed prior to arrival at the IDPS; with initial ingest processing providing Raw Data Records (RDRs).

The RDRs, which contain raw data on a per sensor basis, are subsequently processed to create Sensor Data Records (SDRs), Temperature Data Records (TDRs), and Environmental Data Records (EDRs). In support of the creation of the RDRs, SDRs, TDRs, and EDRs, IDPS is also responsible to collect various Mission Support Data (MSD) used in the system. These are used internally and archived to CLASS.

The IDPS Infrastructure (INF) Subsystem manages the startup and shutdown of all IDPS processes as well as providing common interfaces for all Sis.

The ING SI serves two main functions. The ING SI interfaces with C3S Data Handling Node (DHN) to receive SMD in the form of Extended Application Packets on TCP/IP sockets to produce RDRs. ING SMD strips off the extended headers and stores only the Consultative Committee for Space Data Systems (CCSDS) Application Packets (APs) into the RDR. ING creates metadata and stores the RDRs into the Data Management Subsystem (DMS).

ING also receives Ancillary and Auxiliary files referred to as Mission Support Data (MSD) on specific directories called “Landing Zones”. These files are read from the landing zone and stored in the native format to DMS with metadata. ING MSD receives both dynamic data during normal operations and static data received from the Integrated Support Facility (ISF) during software updates. In some cases, ING also converts the received “native formats” to an “internal format” to provide quicker access for the Processing (PRO) SI. These “internal formats” are also stored to DMS with appropriate metadata.

After products are stored to DMS they are delivered to both internal users and the external community by the Data Delivery Subsystem (DDS). One of the internal users of JPSS products is the Data Quality Monitoring Subsystem (DQM).

In addition to the JPSS constellation of satellites, JPSS CGS receives Application Packet Identifier (APID) Sorted Data (ASD) files and Two Line Element Set (TLE) from the Japanese Aerospace Exploratory Agency (JAXA) Global Change Observation Mission (GCOM) Water satellite. The ASD files contain packets which are processed into RDRs and delivered along with the TLE to CLASS by DDS.

1.3 DOCUMENT OVERVIEW

This document describes the operation of the ING Subsystem software.

Section 1 – Basic overview of IDPS and ING
Section 2 – References used in this document
Section 3 – Software summary including Applications, SW organization, performance characteristics and interfaces to other systems
Section 4 – Basic concepts including startup/shutdown, user interfaces, and data backup
Section 5 – Reference guide including definitions and concepts
Section 6 – Messages and Errors

2.0 REFERENCE DOCUMENTS

TABLE 2.0-1 REFERENCED DOCUMENTS

Document Number	Title
UG60917-IDP-029	IDPS ING SW Config Guide User Manual
UG60822-IDP-011	IDPS INF SW User's Manual Part 2
UG60917-IDP-024	IDPS INF C++ SW App Pro Interface (API) Guide
UG60917-IDP-015	IDPS DMS Software User's Manual Part 2
UG60822-IDP-021	DMS App PGM Interface (API) Guide Build 1.5
474-00014	C3S to IDPS ICD
474-00002-02	CIS ICD Vol II Internal
IC60917-IDP-002	JPSS CGS Data Processor Inter-subsystem Interface Control Document (DPIS ICD) Vol I - IV
474-00001-01	CDFCB-X Vol I - Overview
474-00001-02	CDFCB-X Vol II - RDR Formats
474-00001-05	CDFCB-X Vol V - Metadata
474-00001-06	CDFCB-X Vol VI - Ancillary Data, Auxiliary Data, Messages, and Reports
472-REF-00057	NPP Mission Data Format Control Book and App A(MDFCB)
Drawing 568423 (NPP CDRL 09)	NPP Command and Telemetry (C&T) Handbook
474-REF-00111	Joint Polar Satellite System (JPSS) Ground System Technical Exchange with JAXA for Global Change Observation Mission Water 1 (GCOM-W1)
SGC-070078	GCOM Mission Operations Interface Specification (MOIS)

3.0 SOFTWARE SUMMARY

3.1 SOFTWARE APPLICATION

The Ingest subsystem's function is to bring both SMD and MSD data into the IDPS.

3.1.1 SMD

The ING SMD function receives CCSDS Application Packets from one of two interfaces. To receive NPP data, ING SMD monitors TCP/IP sockets as defined in the C3S to IDPS ICD. Each of these sockets contains specific content filtered by Virtual Channel ID (VCID).

To receive GCOM W1 data (GW1), ING monitors a landing zone directory and receives files of application packets. These files contain packets organized by APID and VCID containing all data from the current downlink from the satellite and the last.

In all cases, the SMD is collected into temporal RDR granules based on the observation time in the packets and stored to DMS.

3.1.2 MSD

The ING MSD function acquires data that has been placed into a series of directories called the Ingest Landing Zone. Each directory expects a specific type of data. The MSD function converts the files to internal format where necessary. Both the original and internally formatted data may be stored to DMS for use by other subsystems.

3.2 SOFTWARE ORGANIZATION AND OVERVIEW OF OPERATION

3.2.1 Logical Components of the Software / Process Architecture

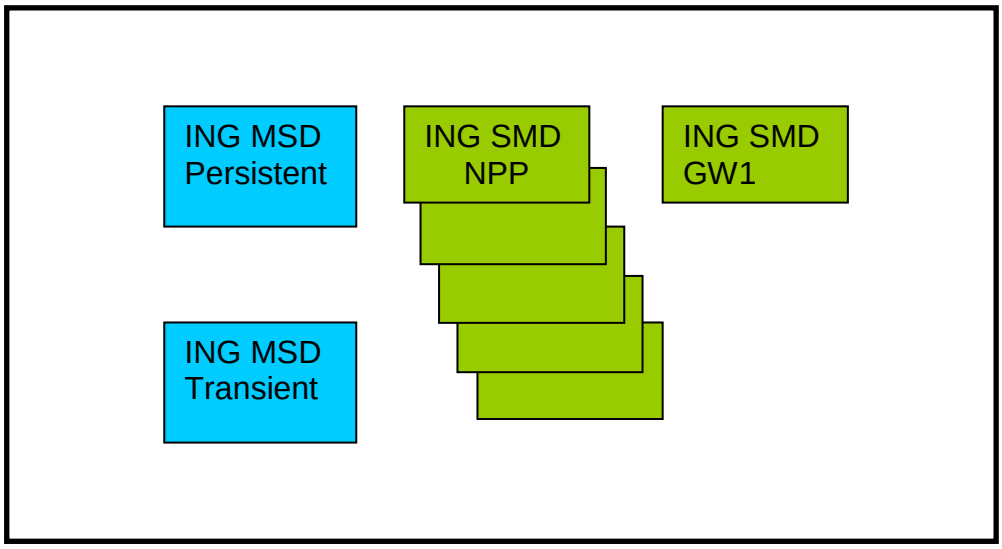

FIGURE 3.2.1-1 ING PROCESS ARCHITECTURE

3.2.1.1 SMD

The SMD component of ING is a single executable. In order to build RDRs, multiple instances of the executable are used to balance the workload based on the bandwidth of the various data streams. Each instance is configured using startup parameters and configuration guide values to identify its task. Each instance monitors a subset of the total data for a specific spacecraft.

3.2.1.2 MSD

The MSD component of ING is a single executable. This executable is configured as one of two processes. The first configuration is persistent and processes all dynamic MSD during normal operations. This includes dynamic ancillary and auxiliary data. The second configuration is transient and is used only during a “software update” to bring in static MSD such as climatology and algorithm lookup tables.

3.2.2 Performance Characteristics

3.2.2.1 SMD

3.2.2.1.1 C3S Socket Interface

SMD processes data as a continuous stream of Extended Application Packets (EAPs) on a suite of TCP/IP sockets. Each EAP can be up to 65,582 bytes in size. The rate of the data expected from C3S is defined in the C3S to IDPS ICD. RDR generation occurs on a data-driven basis that varies based on the temporal size of the RDR granule being generated. Due to the lack of availability of data to C3S between contacts, it is possible that for a time no RDRs may be created. In general, if data is sent to the socket, ING creates RDRs and logs messages accordingly.

3.2.2.1.1.1 Socket Buffer Size Tuning

It has been observed that send and receive buffers that are configured too small can hinder the latency and availability between C3S and IDPS ING. This is mainly due to periods where ING is doing things other than reading the sockets. This can cause a backup to C3S DHN. In this case, the sockets sizes defined are too small and should be increased.

3.2.2.1.1.2 Round Robin Socket Approach

ING processes monitor one to many sockets based on the process configuration. The SMD processes typically listen to two sockets for each VCID: first-copy and retransmit. The first-copy socket is the normal stream of data passed through C3S from the satellite download. If C3S determines that some data was missing, they will retrieve the missing data and broadcast it to IDPS as retransmit data. IDPS ING will request retransmit data each time a FC socket has to be re-established or if it determines an RDR is corrupt. The IDPS Operator can make requests from the IDP Operator GUI if it is deemed necessary as well. This retransmit data is sent down the retransmit sockets. Review Section 5.1.3.1.1.3 for further details on ING retransmit requests.

The sockets are monitored in a round robin approach where each socket (first copy and retransmit) is looked at individually to see if there is data. If any socket is empty, ING checks the next and so on. Since data can be on any of those sockets at any time, it is necessary to limit the number of VCIDs that any socket monitors due to this overhead. If at any time, a socket is no longer connected to its data provider, ING attempts to re-establish the connection.

3.2.2.1.1.3 Missing sockets

Since C3S could send data at anytime on any socket, ING must be connected to all sockets at all times. C3S maintains retransmit sockets all the time, but during a non-nominal situation or

during testing, it is possible that these sockets would not be available. This should be avoided if at all possible because ING tries to reconnect to the socket during every read attempt. This, in turn, slows the SMD process down when compared to typical socket reads. If not having a retransmit socket is desired due to testing, the SMD process can be configured to not establish these sockets as shown in the ING SW Configuration Guide User Manual.

In much the same way that a missing retransmit socket can degrade performance, so can any socket that ING expects to listen to. As the sockets are distributed across multiple hardware systems on the C3S side, it is possible for just a subset of the available sockets to go down at any time. As such, either first-copy or retransmit sockets could go down during processing, causing the same degradation. The use of backups and redundancy within C3S is designed to minimize this risk, but care should be taken during testing to ensure all sockets that a process expects will have a socket available from the data provider..

3.2.2.1.1.4 Keep Alive settings

Unfortunately one of the weaknesses of TCP/IP is that it is possible for the server side (C3S) to go down abruptly in such a way that the client (ING) does not get a signal of the connection loss. When the server comes back up, it waits for a client connection that never comes since the client is still connected to a ghost connection. In order to minimize this issue, ING SMD sockets are implemented with the capability to enable KeepAlive.

This setting initiates a timeout for when no data has been received for a period of time. At that time, it sends a probe to the server to see if it is alive. After a configurable number of attempts, the socket is torn down automatically and ING reconnects to the DHN. This minimizes the downtime due to a ghost connection.

This functionality can be enabled/disabled based on the configuration guide parameter “ING.SMD.<satellite>.SOCKET.KEEPALIVE”. Additional settings to determine the responsiveness include:

“ING.SMD.<satellite>.SOCKET.KEEPALIVE_IDLE”
“ING.SMD.<satellite>.SOCKET.KEEPALIVE_INTVL”
“ING.SMD.<satellite>.SOCKET.KEEPALIVE_CNT”

For additional details see the ING SW Configuration Guide User Manual

3.2.2.1.2 GCOM File Interface

There are minimal performance issues with the GCOM file interface as it is low volume and only minimally coupled to the provider. The only real performance issue is that the file organization is not always beneficial to RDRs that have more than one APID assigned to them (there are two). The file organization forces the RDRs to have longer timeouts to support processing basically two orbits of data (current and last) of one APID before getting the next. The timeouts are lengthened to support holding the RDRs open to minimize versioning.

3.2.2.1.3 Asynchronous Writes

As packets are received by ING, it is very important that a majority of SMD processing time is spent reading data and copying the packets into an RDR. One of the largest impacts to this is the time spent persisting an RDR to disk. As such, final storage of RDRs can be done using separate threads managed by DMS. When an RDR is complete, it is released to the DMS API flagged for final storage. This final storage is handled by the DMS API asynchronously using multiple threads. This asynchronous storage includes the messaging that INF uses to trigger the SDR production. Because multiple threads can finish at different speeds, it is possible that log messages will arrive in different order than the linear granule ID order. The maximum number of threads is configurable per SMD process instance. There is also a flag to turn off asynchronous writes for all SMD processes if desired. Both configurable parameters are defined in the ING Configuration Guide User Manual.

3.2.2.1.4 Shell RDRs

It is possible that ING could receive duplicate packets. These duplicate packets will generally be dropped by the ING SMD process. There is a performance issue if the duplicate data is for an RDR that is 100% already. This is due to the fact that ING is data-driven and for each duplicate packet received, the SMD process will search its list first, and then query DMS if not found. This process of querying DMS for every packet is not very efficient. Due to this latency hit, ING has implemented a concept of a shell RDR in its RDR list when data is received for a complete RDR. This shell acts as a placeholder in the RDR tracking list to minimize the number of queries to DMS. This shell is never created as an RDR object or in DMS, it is just an entry in the list.

The shell entry stays in the list for a configurable time based on the RDR Full Timeout as documented in Section 5.1.7.1.3.4.

3.2.2.2 MSD

The ING MSD process polls a series of landing zones periodically. This period is based on a configurable entry in the ING MSD configuration guide. The actual time it takes for a file to be ingested after being placed in the landing zone is dependent on a number of factors. These include the number of other files in the various landing zones, the size of the file in question, the format of the file in question, and the period of polling. Thus the latency and data rate are indeterminate. There are no latency requirements for ING MSD processing, however. Any MSD file located is ingested by the process and released to DMS as natively formatted data, internally formatted data, or both.

3.2.2.2.1 Software Update as offline process

The transient configuration of the MSD process is meant to be run as an off-line process since it can be a long and resource heavy endeavor. It should only be run when the PRO SI is not executing its processes to minimize the hardware impact. This also allows all PRO processes to grab the correct version of the ingested files.

3.2.3 Interfaces to Other Systems

The primary interfaces for both ING Software Components are DMS and INF. Both use DMS to store their respective products for later retrieval by itself and other subsystems. They also interface with INF to provide them with access to messaging and other needed utilities. Detailed information about these interfaces can be found in the respective User's Manuals for both SIs. INF interfaces are defined in the IDPS INF C++ SW App Pro Interface (API) Guide and IDPS INF SW User's Manual Part 2. DMS interfaces are described in DMS App PGM Interface (API) Guide Build 1.5 and IDPS DMS Software User's Manual Part 2.

The primary interfaces unique to SMD are those linking it with C3S via the DHN. This interface supports the transfer of SMD from the C3S system to the DPE through ING. In addition, the SMD process makes use of the Granule ID utility from INF to provide unique deterministic identifiers for all RDRs, the DNS Utility to retrieve connection information when using the socket interface, and the SMD Retransmit API to request SMD that was lost during an outage. Additionally, SMD also uses an interface with DDS to receive GCOM SMD files

The primary interfaces unique to MSD are those linking it with DQM, DDS and the ISF. The interface with DQM and DDS provides dynamic MSD to the ING landing zone. ING MSD also interfaces with the ISF to receive files used during the software update process. Additionally, for any Landing Zone directory configured for Checksum Verification, the INF Checksum API is utilized.

The following diagrams document the various interfaces that ING uses.

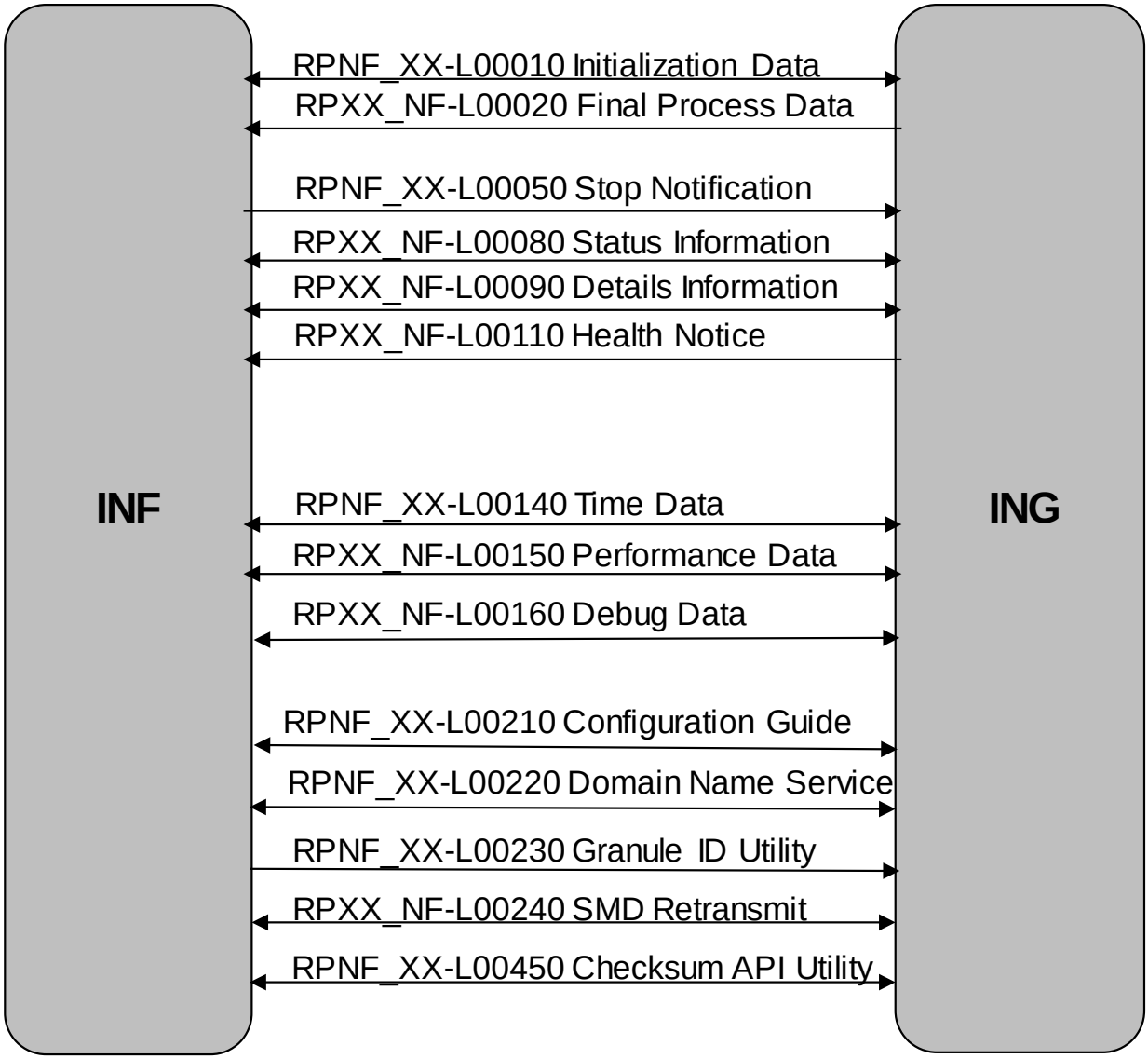

FIGURE 3.2.3-1 INF TO ING INTERFACES

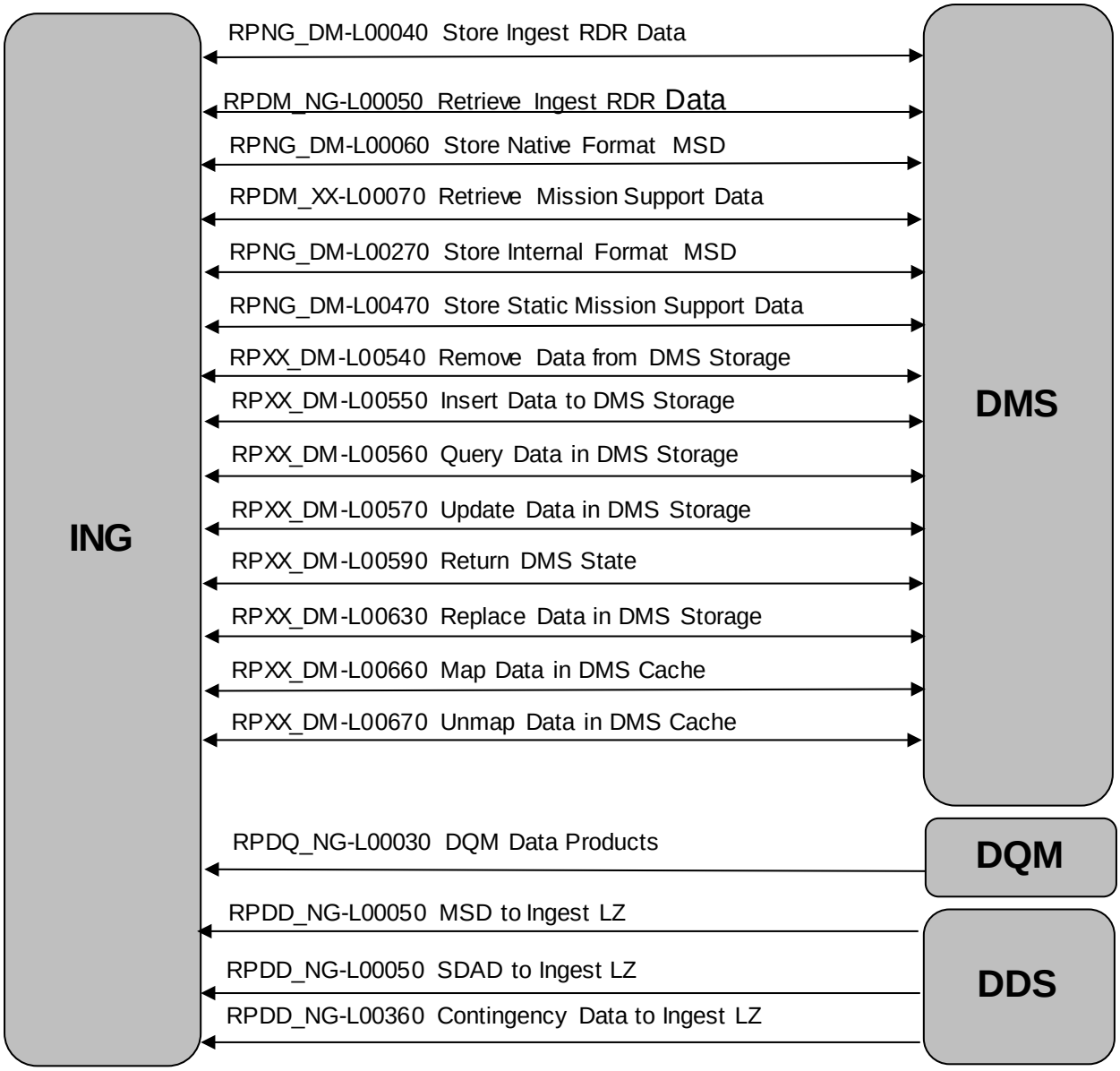
ING
DMS
DQM
DDS
RPNG_DM-L00040 Store Ingest RDR Data
RPDM_NG-L00050 Retrieve Ingest RDR Data
RPNG_DM-L00060 Store Native Format MSD
RPDM_XX-L00070 Retrieve Mission Support Data
RPNG_DM-L00270 Store Internal Format MSD
RPNG_DM-L00470 Store Static Mission Support Data
RPXX_DM-L00540 Remove Data from DMS Storage
RPXX_DM-L00550 Insert Data to DMS Storage
RPXX_DM-L00560 Query Data in DMS Storage
RPXX_DM-L00570 Update Data in DMS Storage
RPXX_DM-L00590 Return DMS State
RPXX_DM-L00630 Replace Data in DMS Storage
RPXX_DM-L00660 Map Data in DMS Cache
RPXX_DM-L00670 Unmap Data in DMS Cache
RPDQ_NG-L00030 DQM Data Products
RPDD_NG-L00050 MSD to Ingest LZ
RPDD_NG-L00050 SDAD to Ingest LZ
RPDD_NG-L00360 Contingency Data to Ingest LZ

FIGURE 3.2.3-2 DMS/DQM/DDS TO ING INTERFACES

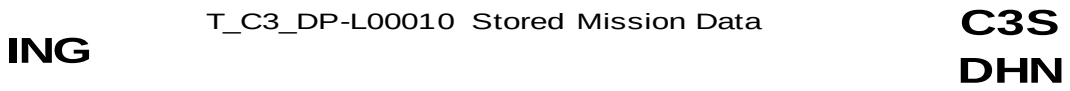

FIGURE 3.2.3-3 C3S TO ING INTERFACE

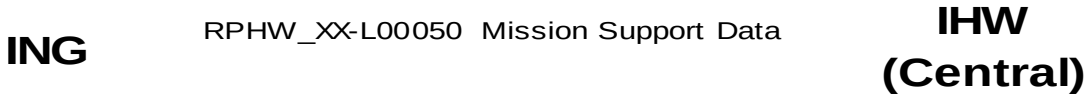

FIGURE 3.2.3-4 SOFTWARE UPDATE INTERFACE

3.2.4 Control

The ING Subsystem requires no additional passwords or controls in addition to those generally needed by IDPS.

3.3 ALTERNATE MODES OF OPERATION

3.3.1 Lightweight mode

All ING processes can be started in a “light-weight” mode for testing purposes either in the Clearcase VOB or within a deployed environment. In this mode, the process is started with a command line argument string starting with the process name followed by “-lw <tk config file>” followed by the normal startup parameters for the respective process. The “<tk config file>” is an XML file that provides information normally provided by the IDPS GUI process startup. It includes logging modes, directories where logs should be written, and tasking information. In light-weight mode, the IDPS GUI is not used and all status and detail messages go to files.

For additional information on light-weight mode see the INF Software User’s Manual Part 2.

3.3.2 SMD

3.3.2.1 Retransmits

SMD processes can be configured to perform retransmits or not. This is accomplished by changing the configuration guide parameter ING.SMD.<satellite>.RETRANSMIT from “true” to “false in the specific satellite configuration guide. Operationally, the ING SMD processes receiving data from C3S are configured to receive data on both a First-copy (FC) and a Retransmit (RETR) socket for each VCID. When retransmits are disabled, the process will not connect to the RETR sockets in addition to not doing retransmit requests. This can be used for simplified testing.

The GCOM SMD process must always have retransmits disabled as there is no interface for IDPS to request lost data.

For additional details on this Configuration Guide item see the ING Configuration Guide User Manual.

3.3.2.2 Extended Headers

The ING SMD processes with C3S are normally configured to receive Extended Application Packets on the sockets they are monitoring. Extended Application Packets contain additional header information added by C3S. The SMD process can be configured to receive Application Packets if desired. Toggling between AP and EAP is accomplished by changing the configuration guide parameter ING.SMD.<satellite>.EXT_HEADER from “true” to “false” in the specific satellite configuration guide. When in this mode, the SMD process doesn’t look for the extended header as it reads the packets from the socket. If the configuration of the SMD process does not match the data stream, expect multiple types of errors.

The GCOM SMD processes must always have extended headers disabled as KSAT does not provide them.

For additional details on this Configuration Guide item see the ING Configuration Guide User Manual.

3.3.3 MSD

3.3.3.1 Persistent Mode

The persistent mode for MSD is the standard mode of operation. It runs the ING process as a daemon process. A process started in this way should keep processing for the duration of an IDPS instance's lifetime. This mode is used to process any dynamic MSD, both ancillary and auxiliary.

3.3.3.2 Transient Mode

The transient mode of operation for MSD is used during the software update process. Its goal is to process files that are only periodically brought into the DPE. It examines the landing zones listed in section 5.2.2 one at a time, processing any MSD present. Once that is done, the process terminates.

3.4 SECURITY

There are no security considerations for ING processes. ING processes are only started using the IDP Operator display which provides the needed security protocols.

4.0 BASIC CONCEPTS

4.1 STARTUP AND SHUTDOWN

4.1.1 Startup

ING processes are started using the IDP Operator Display. If the user wants to start all of ING at once, they can start the ING Group from the Group tab. This starts all but the "ING MSD – Static" which is only used for software updates and the "ING SMD GW1" process which is only utilized at NESDIS and can be started with the "ING SMD GCOM Group". The following sections define how to start up the individual processes of the ING subsystem from the IDP Operator Display.

Additional details of process startup can be reviewed in the INF SW User Manual Pt 2.

4.1.1.1 SMD

4.1.1.1.1 Processes

From the Processes tab, press the New Process button. A drop down list of available process choices is presented. ING SMD has one choice for each task type, generally organized along spacecraft and sensor lines. Choose the SMD process corresponding with the data set you are going to process. The options are documented in the following table:

TABLE 4.1.1.1.1-1 SMD PROCESS CONFIGURATIONS

Process Name	Description
ING SMD ATMS	Processes NPP ATMS Science data
ING SMD ATMS DIAG	Processes NPP ATMS Diagnostic data
ING SMD CERES	Processes NPP CERES Science/Diagnostic data
ING SMD CrIS	Processes NPP CrIS Science data
ING SMD CrIS DIAG	Processes NPP CrIS Diagnostic data
ING SMD OMPS	Processes NPP OMPS Science data
ING SMD OMPS DIAG	Processes NPP OMPS Diagnostic data
ING SMD VIIRS	Processes NPP VIIRS Science data
ING SMD VIIRS DIAG	Processes NPP VIIRS Diagnostic data
ING SMD Sensor Dump/Dwell	Processes NPP Sensor Dump/Dwell data
ING SMD Sensor/SC TM	Processes NPP Telemetry data for all sensors and the spacecraft
ING SMD GW1	Processes GCOM W1 data (NESDIS Only)

4.1.1.1.2 Modes

ING SMD processes can be started up with either a “cold start” or “warm start” mode. The only difference is that during a “warm start” when connected to C3S, the ING process will request data that was missed while it the sockets were disconnected. This is known as a retransmit request. Detailed description of the Retransmit Request interface can be found in section 5.1.3.1.1.3. During a “cold start”, the gap is ignored and no requests are made. After the initial startup, these two modes are identical.

The GCOM SMD process has no distinction between “cold” and “warm start”.

For additional info on configuring retransmit request see section 5.1.3.

4.1.1.2 MSD

4.1.1.2.1 Processes

From the Processes tab, press the New Process button. Then, a drop down list of available process choices is presented. Choose the MSD process corresponding with the data set you are going to process. The options are documented in the following table:

TABLE 4.1.1.2.1-1 MSD PROCESS CONFIGURATIONS

Process Name	Description
ING MSD – Dynamic	Processes dynamic data when IDPS is in operational mode
ING MSD – Static	Processes static data when IDPS is in a software upgrade mode

4.1.1.2.2 Modes

The MSD process has no distinction between “cold start” and “warm start” modes.

4.1.2 SHUTDOWN

Select the Processes tab to bring up a list of processes that are running. Select the process you would like to shut down from the list of running processes confirm you wish to shut it down.

Additional details of process shutdown can be reviewed in the INF SW User Manual Pt 2.

4.2 USER INTERFACE OVERVIEW

Ingest has no user interface of its own. All messages are relayed through the INF subsystem and appear either in logs or within the IDP Operator display.

4.3 RELATED PROCESSING

None

4.4 DATA BACKUP

None

5.0 REFERENCE GUIDE

5.1 SMD

The function of ING SMD is to capture application packets into RDRs, track completeness of the SMD data in the RDRs, and store them to DMS.

5.1.1 Startup parameters

As noted previously, there are multiple instances of a single executable used for all SMD processing needs. This executable receives its tasking at startup from process arguments. These process arguments are effectively hidden from the user when starting up the SMD processes using the IDP Operator Display. There are two arguments designated with flags of –ingSC and –ingPROC. The first argument is the spacecraft that the SMD originated from.

The second argument is the configuration number for the process. It is in the format “ING1”, “ING2”, etc. There are a finite number of these configurations for each spacecraft. Each of these configuration numbers has a defined list of RDRs that they create as defined in the respective ING Configuration Guide for each satellite. The current configurations are documented in Table 5.1.1-1.

TABLE 5.1.1-1 SMD STARTUP PARAMETERS

Spacecraft	Configuration	Description
NPP	ING1	Processes NPP VIIRS Science data
NPP	ING2	Processes NPP ATMS Science data
NPP	ING3	Processes NPP CrIS Science data
NPP	ING4	Processes NPP OMPS Science data
NPP	ING5	Processes NPP CERES Science data
NPP	ING6	Processes NPP VIIRS Diagnostic data
NPP	ING7	Processes NPP ATMS Diagnostic data
NPP	ING8	Processes NPP CrIS Diagnostic data
NPP	ING9	Processes NPP OMPS Diagnostic data
NPP	ING10	Processes NPP Sensor Dump/Dwell data
NPP	ING11	Processes NPP Telemetry data for all sensors and the spacecraft
GW1	ING1	Processes all GW1 data

Ignoring the startup mechanism (light-weight vs INF), the following would be used to start the ING1 configuration for the NPP spacecraft:

IngSmd.exe –ingSC NPP –ingPROC ING1

5.1.2 Application Packets

ING SMD receives APs containing a primary header which provides identifying information. This includes things like the Application Packet ID (APID) and segmentation flags. There is also a secondary header in packets that contains observational timestamps. The rest of the packet is the user data content.

Content details of application packets for NPP can be found in the NPP Mission Data Format Control Book (MDFCB) or the NPP Command and Telemetry (C&T) Handbook.

Content details of application packets for GW1 can be found in the JPSS GS Technical Exchange with JAXA for GCOM-W1 document.

5.1.3 Data Provider

5.1.3.1 C3S interface

The SMD processes receive data on TCP/IP sockets in the form of Extended Application Packets (EAPs) as defined by the C3S to IDPS ICD.

5.1.3.1.1 Extended Application Packets

ING SMD receives EAPs from C3S. EAPs are Application Packets (APs) that have two C3S headers added to the front of the packet. They are the Extended AP Header and the Extended VCDU Header. These extended headers are removed from the AP when stored to RDRs, but information about the amount of fill in the AP is persisted in the RDR and utilized for percent complete calculations.

The extended headers are defined in the Internal Data Format Control Book (IDFCB) Volume 1.

5.1.3.1.2 Socket Connections

Based on the parameters provided to the SMD process at startup, it can determine what RDRs it is building. This is determined by retrieving a list of RDRs from the ING configuration guide for the specified ING process (see Startup Parameters). The process can access what Application Packet Identifiers (APIDs) it is monitoring to create these RDRs. Each APID is described in detail in the configuration guide including the maximum size, frequency and the Virtual Channel ID (VCID) on which the data is sent.

The SMD process establishes TCP/IP connections for each VCID. Every VCID identified has its own connection to the data provider to receive packets. If retransmits are enabled, the process establishes two connections to the data provider: First Copy and Retransmits. If not,

only a First Copy connection is made. First copy data is data received from the data provider in a nominal path from the satellite. Retransmitted data is received in a non-nominal path from the data provider when a retransmit request triggered the event. This request could be a result of a lost contact during a downlink, network failures, or hardware failures with the data provider or IDPS that required the request be made. Retransmit sockets are only established if C3S is the data provider.

A single SMD process monitors one to N sockets depending on the configuration. These sockets are accessed in a round-robin manner, processing one packet per socket at a time as available. If no data is available on a given socket, the next socket is accessed, until the process is shutdown. If no data is available on any sockets, a message is periodically sent that “No Data is available”. If the process receives a configurable number of invalid application packets on a specific socket, it disconnects from the data provider and attempts to re-establish a new connection.

Additional information about socket connections can be found in the C3S to IDPS ICD.

5.1.3.1.2.1 Domain Naming Service

In order to establish sockets, ING utilizes the INF DNS Utility to provide IP address and port numbers for the data provider sockets. The data provider posts a key that the ING process uses to request the server and port information. This DNS key is assembled dynamically using configuration guide parameters and other startup details. Once the DNS key is assembled, and the address of the data provider is received, the process attempts to connect. If a socket connection fails, the process continues attempting to connect until either it is successful, or the process shuts down. Once a connection is established, packets can then be read. The key itself looks like the following:

_<SMD_MODE>._DHN_NA_SMD_<SATELLITE>_<RETRANSMIT_TYPE>_<VCID>_<SITE_NAME>._tcp

Additional details about DNS keys can be found in the Common Interfaces and Services ICD. Inputs to the DNS key can be found in Table 5.1.2.2-1.

TABLE 5.1.2.2-1 DNS KEY INPUTS

Field	Description
SMD_MODE	Retrieved from configuration guide value ING.SMD_MODE
SATELLITE	Received as part of the startup parameter for the process (see Startup Parameters)
RETRANSMIT_TYPE	Based on the socket being established. Either FC (First-copy) or RETR (Retransmit)
VCID	Based on the socket being established
SITE_NAME	Retrieved from configuration guide value DPE.SITE_NAME

5.1.3.1.3 Retransmit Requests

Retransmit requests are performed to support Data Availability. If configured to perform them, ING can request data from C3S that it has determined to be lost. These requests are made whenever a FC socket is re-established or if ING determines that an RDR is corrupt. The requests are performed using the INF SMDRES API that creates an SMD Retransmit Request message that is processed by an INF back end process. This process communicates with C3S to make the requests. Requests are based on a temporal period for a specific VCID.

When a socket is re-establlshed due to a disconnect, ING will determine the temporal period that must be requested. The temporal period is based on maintaining the port state (time of the last packet received) of the FC socket. This state is used as the basis for the start of the request and is kept in memory and persisted to a port state file periodically and at shutdown. There are four scenarios that determine the range of the requests for socket re-connect.

The first type of request is simply based on the time of the last packet received (from the port state) to the time of the first packet received after reconnect. The second scenario is utilized when no data arrives on the socket within a configurable amount of time. This could occur if the program is between contacts. In this case, ING will request from the last packet time to a symbolic “now”. This is accomplished using a flag in the message to C3S. In this case, C3S will decide what “now” is based on their most current data for that VCID.

The final two types for socket re-connects are based on extended outages. If an outage is too large, it is possible that there will not be adequate down time to catch up with all data lost without impacting latency on other granules. In the third type of request, ING evaluates the port state and if it is older than a configurable value, ING will only requests from the first packet received time less the maximum to the first received time. The final request of this type is the same logic but for when no data arrives for a configurable time. In this case, ING requests the current time less the maximum request to the symbolic “now”.

The final type of retransmit request is based on ING receiving data for an RDR and finding that RDR is corrupt when retrieved from DMS. This can occur if ING is working with an RDR and

abruptly and ungracefully dies (i.e. power outage). Since the RDR was in memory, it doesn't get persisted to disk which is the error ING receives when retrieving it. ING deletes the RDR from DMS to cleanup the inventory and then requests the temporal range of the RDR to ensure that data is re-sent to create the RDR.

5.1.3.1.3.1 Port State Files

To properly identify the gap for a retransmit request, ING must maintain a state of the socket, information describing the last AP received. This is maintained in memory and persisted to a binary file, known as a Port State File, periodically and every time the process is shutdown. This enables ING to make a retransmit request at startup if desired. ING SMD maintains one of these files for each of the VCIDs that it is listening to.

The format of these files is **ING_SMD_StateFile_<*satellite*><*VCID*><*domain*>**. For example, **ING_SMD_StateFile_NPP0011OPS**, would be a valid filename.

All of these state files simply contain the observation time value of the last application packet received. Due to the possibility of packets with unusable timestamps, ING will not update the port state with a timestamp that is less than the Granule Base time for the particular spacecraft (these packets are dropped anyway). Port State files are located in a configurable directory identified in the ING Configuration Guide.

See the ING Configuration Guide User Manual for more information.

5.1.3.2 GCOM

The GCOM interface to receive application packets is different than the C3S one. It receives files in a landing zone where they are parsed to create RDRs.

5.1.3.2.1 File-based

GCOM files are organized as one file per APID per VCID for each contact received at Svalbard. Each file actually contains the current contact as well as the previous to serve as a backup. ING SMD processes the files one at a time, extracting packets one at a time from the file to process into the correct temporal RDR.

The packets are standard CCSDS format packets with no extended headers.

5.1.3.2.2 Landing Zone

GCOM processing is supported with two directories: GCOM_ASD and GCOM_ASD_TEMP. The first directory receives all the packet files delivered from the I-MSDS. The TEMP directory is a holding area used by ING for temporary storage of files that ING determines there are

issues with. These directories are located based on the “ING_MSD_LZ_PATH” environment variable.

Files are placed in the GCOM_ASD directory using the native JAXA filename defined in the GCOM-W1 Mission Operations Interface Specification (MOIS). The files are copied into the directory by DDS SW on the I-MSDS using a temporary naming convention methodology by appending “.TMP_FILENAME” as an extension to the filename. This flags the file to ING as being in transfer. When the extension is removed and the file only has an “.L0D” extension, ING can process it. If a file has been in the landing zone longer than a configurable length of time with the temporary name, it is moved to the TEMP directory where the operator can review it.

ING SMD processes each file one packet at a time into RDRs. As each file is complete, it is deleted from the landing zone and the next one is read. If a file is determined to have too many errors (invalid APIDs, etc), ING will move the file to the temporary holding directory and a new file is read and processed.

Periodically the SMD process will review the TEMP directory and determine if any files have been there longer than a configurable period of time. For each file that is older than this configurable time, ING removes them from the directory.

5.1.4 Drop Packets

ING SMD verifies that the APIDs it receives are correct on a per VCID basis. If it receives invalid APIDs, it logs the error, drops the packet and continues with the next packet. The APID to RDR assignments are documented in the CDFCB-X Volume 2. There are certain APIDs that are not assigned to RDRs, but will arrive regularly in the data stream. These include packets that are used as fill in the downlink and packets that IDPS cannot process due to their lack of a secondary header timestamp.

This function is accomplished by configuring ING to know the specific APIDs to drop so that they are not handled as Invalid APIDs. The APIDs to drop are configured in the satellite-specific configuration guide for ING. These are logged using the DROP_PKT message like all dropped packets to make the trackable.

See the ING Configuration Guide User Manual for more information.

5.1.5 Packet Timestamps for Segmented Packets

In order to assign packets to an RDR, ING must be able to identify a timestamp for the packet. In the case of segmented packets, where a group of packets are considered a single unit in CCSDS terms, only the first packet of the segment is required to have a timestamp. The continuation packets do not always contain a secondary header timestamp, making this identification difficult. ING assigns these continuation packets a timestamp to be used for identifying to the correct RDR. This is done through the use of an internal tracking mechanism.

When ING receives a packet, it determines if the packet is standalone or segmented. If the packet APID is segmented, ING looks at the Primary Header Group flag to see if it is the first, middle or last packet of a segment. If the packet is a first, the tracking array is updated with the new timestamp for the segmented group and the number of packets for the segment is extracted from the secondary header of the packet. This number of packets is used to assign a range of sequence numbers to the tracking array that is valid for the segmented group. As continuation packets arrive with sequence numbers in the established range, they are assigned the first packets timestamp and forwarded to RDRs.

If a packet is received with a sequence number that is not in the current range and it is not the first packet of a new segment, the packet is dropped as “indeterminate”. This dropped packet is logged for traceability using the DROP_PKT message.

5.1.6 Timestamp Delta Modification

Sensors build packet timestamps in several ways. In most cases they are either the start of the scan of the instrument or the time of observation of the data. These are very simple for ING and later processing algorithms to work with. There are circumstances, however where the timestamp may be after the data was collected and encoded into the application packet. This can be an issue for later processing if data needs to be coordinated with another data source.

In order to facilitate this coordination, ING SMD has the capability to use a delta to the timestamp in a packet to place it in the RDR. This does not change the physical packet time, but will align the data into a different RDR granule so that the granule ID of the RDR will align to the real observation time, even if the time in the packet itself does not. The delta time is configured per APID in the satellite specific ING Configuration Guide. ING will only perform delta timestamp modification for APIDs listed in the configuration guide. The configuration guide may include several deltas per APID if the deltas change over time. This allows processing of older data with the known delta of that time.

See the ING Configuration Guide User Manual for more information.

5.1.7 RDRs

5.1.7.1 Common Concepts

5.1.7.1.1 Sizing and structure

An RDR is a binary structure containing application packets received for a given type of data. RDRs are organized along sensor and functional lines, so VIIRS Science may be one RDR category, while CrIS Diagnostic would be another. The binary structure of an RDR consists of four parts; the static header, the apid list, the packet tracker, and the data section

The static header section contains data pertaining to the RDR as a whole. This includes things like the satellite, sensor, RDR type, granule boundaries, and byte offsets into the other sections. The static header is the same size for each RDR, and is used as a reference point for the other sections.

The APID list section contains a list of APIDs in the RDR with their symbolic name, APID number, the number of packets reserved and received, and an index into the packet list. Basically this section is a list of APID attributes along with a link into each APIDs packet listing in the next section.

The next section is known as the Packet Tracker. This section contains a record of each packet received along with its observation time, sequence number, the packet size, its offset into the storage section, and its fill count. This section is basically just a database of each packet in the RDR.

Finally there is the storage section. This section is where the actual APs are. As each AP comes into SMD, its extended header is removed (if applicable) and the resulting packet is stored “as is” in the storage section. Packets are stored end to end as received.

More detailed information on the RDR format can be found in the CDFCB-X Volume II.

5.1.7.1.2 Granule IDs

ING SMD collects application packets into their respective RDRs based on temporal boundaries. These boundaries are defined by the INF Granule ID utility based on a base time and a temporal granule size. The Granule ID utilizes a granule size in microseconds and “chunks” from the base time to the time of an application packet. This provides the start and end boundary times for the RDR granule. The utility also provides a unique granule ID based on these temporal boundaries.

More detailed information on the Granule ID can be found in the CDFCB-X, Volume 5.

5.1.7.1.3 RDR Timeouts and Thresholds

Within a nominal operating environment, ING receives packets in order, such that RDRs are created, filled, and stored to DMS then the process continues for the next RDR. For various reasons including mode changes, end of contact, and simple loss of data, ING may not receive all data for a single RDR. If this is the case, then after a period of time ING has to release the RDR so that at a minimum DDS can deliver it. The analysis to release incomplete RDRs is accomplished using timeouts and thresholds. Timeouts are used to trigger an analysis of the RDR for completeness and are based on when the RDR object was created (either new or as an update).

Additional detail about Timeouts/Thresholds can be found in the Repair Granules section of the CDFCB-X Volume I.

5.1.7.1.3.1 Primary Timeout/Threshold

The Primary Timeout and Primary Threshold work as a pair and are based on a reasonable amount of time to receive all the data for a given granule during nominal operation. If an RDR is older than the value of the Primary Timeout, the RDR is analyzed to see if the percent complete is greater than the Primary Threshold. If it is, it is released to DMS (unlocked) and messages inform INF it is ready to be processed. RDRs that don't meet this threshold are held in the ING list until they either get all their data or exceed the Unlock timeout.

Primary Timeouts and Thresholds are implemented in pairs and are configurable for each RDR as documented in the ING Configuration Guide User Manual.

5.1.7.1.3.2 Repair Timeout/Threshold

The Repair Timeout and Repair Threshold are used once an RDR has been released for downstream processing and ING receives more packets for it. In this case, an RDR must be retrieved from DMS that has been previously unlocked and a new version is created from the original (see section on RDR Versioning). This version then uses the second pair of timeout/thresholds. The Repair Timeout is typically much longer than the Primary to give C3S time to recover needed data and forward it on. The Repair Threshold is then used to analyze RDRs in the same way as the Primary is used when the Repair Timeout is exceeded.

Repair Timeouts and Thresholds are implemented in pairs and are configurable for each RDR as documented in the ING Configuration Guide User Manual.

5.1.7.1.3.3 Unlock Timeout

As noted above, an RDR can exceed its timeout, but not contain enough data to meet its threshold. In this case, it is assigned a new timeout designated as the Unlock Timeout. This timeout is there so that RDRs do not remain locked indefinitely in the system, and thus it allows DDS to eventually deliver them. This Unlock Timeout is a relatively long duration to allow RDRs a large window of opportunity to become filled.

If an RDR receives additional data such as during a start of contact, the timeout will revert back to the Primary or Repair timeout that it was using previously so that it can be released in a more timely manner. If the Unlock timeout is exceeded due to no additional data, the RDR is unlocked regardless of the percent complete, but INF is informed to not do any downstream processing.

The Unlock Timeout is common for all RDRs as documented in the ING Configuration Guide User Manual.

5.1.7.1.3.4 RDR Full Timeout

This timeout manages the time that shell RDRs are kept in the list of open RDRs. This timeout provides a trade between hits to DMS and the length of the RDR list.

The RDR Full Timeout is common for all RDRs as documented in the ING Configuration Guide User Manual.

5.1.7.1.4 RDR Versioning

RDRs are versioned every time a previously released RDR is retrieved from DMS due to data arriving for an incomplete RDR. Versioning is performed because the original, incomplete, RDR may have already been used in downstream processing. Versioning is therefore used as a way of marking one RDR different from its predecessors or future versions, even if they all share the same granule ID. The initial RDR version is an ‘A1’. Each time that a released (unlocked) RDR is retrieved and filled with more data, the numeric component of the version is incremented. For example the second version would be ‘A2’, the third ‘A3’, etc.

While in theory there can be an infinite number of versions, it is actually limited by the nature of the AP data. Since C3S can only recover data up to 24 hours, this limits the number of versions that would ever be needed since the Repair and Unlock Timeouts provide long periods for late data to arrive.

Additional detail about Versioning can be found in the Repair Granules section of the CDFCB-X Volume I.

5.1.7.1.5 RDR Completion Calculations

For each RDR, Ingest must determine when it is has received all the APs expected. In general, this means that the ING SMD process tracks the number of APs received for any given RDR. Each RDR must also contain logic to determine how many APs it should expect to receive for each RDR. The percent complete is then calculated based on a ratio of received over expected.

The number of APs expected in a given RDR can be as simple as the number of slots allocated at creation, but it also may need to account for state-related issues such as Day vs. Night mode and/or for packets that may not arrive in a deterministic manner. These issues make the expected number of APs a dynamic number for some RDRs. In such RDRs, the number of expected APs is recalculated whenever it obtains new information that alters the expected number of packets.

While in simple terms the actual number of APs is simply the number that has been received and stored in an RDR, it is also modified based on packets that contain fill data. The percent in the C3S Extended AP header percent is used as an approximation of how much of an AP is actually missing. The sum of all fill percents is used to calculate the equivalent number of packets that are missing. This number is subtracted from the number of APs that have been received so that all packets must arrive and contain no Fill data for the RDR to truly be 100% complete.

The following sections describe in greater detail the special conditions that are accounted for in percent complete calculations for RDRs.

5.1.7.1.5.1 Day/Night

Some instruments are affected by the side of the earth they are on. In some cases, this can mean that only some of the normal data can be sent down. This affects the calculation of percent complete and ING attempts to account for this in the percent complete calculations. Additionally, for some instruments, no data comes down due to this concept, but that would mean no RDR is created at all. Since the day/night is on a per scan basis, some RDRs will be mixed mode as well during the transition.

5.1.7.1.5.2 Short Granules

As noted in the Granule ID section above, granules are temporal products based on a number of scans of a sensor. Due to sensors being mechanical in nature, there can be some fluctuation in scan rates which could create a situation where a granule that is expecting N scans actually only receives (N-1). This concept is known as Short Granules. Some RDRs are more prone to this behavior and the specific RDR code must account for it by tracking the timestamps that have been received and analyzing the list for gaps.

The variability also has one other possible result related to short granules. This is a concept of long granules where the instrument actually scans faster than normal and within a given period of time would receive (N+1) scans. This would be a bad thing as the RDR is only sized for N scans. This possibility is eliminated by shrinking the granule size to only allow the short granules to occur.

5.1.7.1.5.3 Fattened Granules

In some cases, an RDR is fattened to account for the variability of data. This is for special cases where the standard short granule will not work. In these cases, the periodicity of APIDs is reduced in the configuration guide to increase the number of packets expected for an RDR for one additional packet per APID. Where this philosophy is used exclusively (without short granule analysis), the nominal rate for packets will not cause the RDR to be 100%, however, a long granule will. As an example, assume an RDR that nominally expects 10 packets but due to variability, the RDR is sized for 11. In this case, when only the nominal count of 10 arrives,

the RDR will not be 100%. For the RDRs that actually have 11 in them, the percent complete is 100%.

5.1.7.1.5.4 Packets with Fill

When re-integrating the APs at C3S, occasionally some of the pieces (known as Virtual Channel Data Units or VCDUs) do not arrive soon enough and C3S completes the packet with "Fill Data". This fill data replaces all missing bytes with a pattern of "xDEAD". The amount of fill is documented in the C3S Extended AP header and is used by ING to calculate the RDR percent complete by only counting the packet as a partial based on the fill value. The Extended AP header and the concept of fill are discussed in the C3S to IDPS ICD.

5.1.7.1.5.5 Asynchronous Packets

There are some packets that are asynchronous by nature. This means that they do not arrive in a defined temporal pattern. They may be triggered as a result of a non-nominal condition encountered by the satellite (an error) or simply by a commanded special event such as memory dumps. In the case of these types of packets, ING SMD must assume a worst case philosophy in regards to sizing and allow RDRs to timeout as the expectation is that received will not ever equal expected for an automatic release.

5.1.7.1.5.6 Super-Segmented Packets

This concept is currently limited to OMPS data streams. This concept is a enhancement to the normal segmentation of CCSDS packets to allow more than 256 packets to be associated with the same timestamp. Super-segmentation allows related data to have a number of packets equal to the sequence number limit. The OMPS packets contain an additional header after the secondary header that contains info about the super-segmentation that can be used to correlate the data. ING uses this information to determine the number of packets expected.

Additional detail about super-segmentation can be found in the NPP MDFCB.

5.1.7.2 Specific RDRs

ING SMD stores specific APIDs to each RDR product. There is no duplication across RDRs. Aggregation of RDR types for delivery is done by DDS. The only specific code that each RDR has is to calculate percent complete. The following sections describe the concepts that impact these calculations.

For the specific APID assignments, see the CDFCB-X Vol II.

5.1.7.2.1 ATMS Science

The ATMS Science RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.1.1 Fattened Granule

Due to the non-scan based nature of the data and the lack of temporal uniformity in the FOVs, this RDR is fattened. The four APIDs in this RDR are not related in timestamps and therefore not all will need the extra slot created by fattening. This means that there is no situation where the RDRs can be 100% so they will always time out and use thresholds to determine the state to release.

5.1.7.2.2 ATMS Diagnostic

The ATMS Diagnostic RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.2.1 Short Granule

Short granules are possible, but since the packet times are not scan based, but rather based on the observation time of the data within numerous points in the scan, the RDR is simply short on data and released using the Timeouts and Thresholds.

5.1.7.2.3 ATMS Dwell

The ATMS Dwell RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.4 ATMS Memory Dump

The ATMS Memory Dump RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.4.1 Asynchronous Packet

Since memory dump requests are dynamic in size and asynchronous by nature, RDRs must be sized to handle the worst case in regards to the number of APs that can be received. Due to this variability, all thresholds are set to zero so that they are released based on timeouts in all cases.

5.1.7.2.5 ATMS Telemetry

The ATMS Telemetry RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.6 CERES Science

The CERES Science RDR must do a dynamic calculation of percent complete. This is due to the variability of the scan rate.

5.1.7.2.6.1 Short Granule

The CERES Science RDR must account for short granules. As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans, effectively changing the number of expected APs.

5.1.7.2.7 CERES Diagnostic

The CERES Diagnostic RDR must do a dynamic calculation of percent complete. This is due to the variability of the scan rate.

5.1.7.2.7.1 Short Granule

The CERES Diagnostic RDR must account for short granules. As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR is only receiving (N-1) scans, effectively changing the number of expected APs.

5.1.7.2.8 CERES Telemetry

The CERES Telemetry RDR must do a dynamic calculation of percent complete. This is due to the variability of the scan rate.

5.1.7.2.8.1 Short Granule

The CERES Telemetry RDR must account for short granules. As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans, effectively changing the number of expected APs.

5.1.7.2.9 CrIS Science

The CrIS Science RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.9.1 Fattened Granule

Due to the non-scan based nature of the data and the lack of temporal uniformity in the FOVs, this RDR is fattened. This means that these RDRs will have to always time out and use thresholds to determine how to release.

5.1.7.2.9.2 Asynchronous Packet

While not an asynchronous packet in the simplest definition, there is an Engineering packet that only shows up every four minutes for this RDR. Given that granules sizes are not expected to approach that size, the RDR must account for the infrequent nature of the packet when it does its percent complete calculation. This is done by always expecting the packet in all granules.

5.1.7.2.10 CrIS Diagnostic

The CrIS Diagnostic RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.10.1 Short Granule

Short granules are possible, but since the packet times are not scan based, but rather based on the observation time of the data within numerous points in the scan, the RDR is simply short of data and released using the Timeouts and Thresholds.

5.1.7.2.11 CrIS HK Dwell

The CrIS HK Dwell RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.12 CrIS IM Dwell

The CrIS IM Dwell RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.13 CrIS SSM Dwell

The CrIS SSM Dwell RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.14 CrIS Memory Dump

The CrIS Memory Dump RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.14.1 Aysnchronous Packet

Since memory dump requests are dynamic in size and asynchronous by nature, RDRs must be sized to handle the worst case in regards to the number of APs that can be received. Due to this variability, all thresholds are set to zero so that they are released based on timeouts.

5.1.7.2.15 CrIS Telemetry

CrIS Telemetry RDRs account for two dynamic components to calculate the expected number of APs. They are Fattened Granules and Short Granules.

5.1.7.2.15.1 Fattened Granule

Due to there being several APIDs of different production, a simple short granule implementation could not be used. This RDR is modified to size the RDR for an additional packet per APID in order to ensure there is enough storage for the variability.

5.1.7.2.15.2 Short Granule

This RDR implements a short granule analysis per APID in order to estimate the exact number of packets that should be expected. It accounts for the fattened sizing where a short granule is the actual nominal case. This is required, as this RDR is used as an input for all SDRs to create geolocation and is therefore a latency driver.

5.1.7.2.16 OMPS FSW Bootup Status

The OMPS FSW Bootup Status RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.16.1 Aysnchronous Packet

The packet that goes into this RDR is asynchronous by nature and therefore this RDR is only created rarely and the RDR will usually be released based on timeouts/thresholds..

5.1.7.2.17 OMPS Dwell

The OMPS Dwell RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.18 OMPS Memory Dump

The OMPS Memory Dump RDR is designed to handle the dynamic nature of memory dumps.

5.1.7.2.18.1 Asynchronous Packet

Memory Dump packets are a commanded data. They do not show up regularly. When a request is made of the satellite, it responds by sending down a memory dump stream as a response.

5.1.7.2.18.2 Super-Segmented Packet

Since the OMPS instrument fulfills memory dumps as a super-segmented packet, the RDR is sized to handle the largest byte request possible. When the RDR receives the first packet of the segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. When the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING can now calculate the exact number of APs it should expect. Until both of these pieces of information is received, the RDR will expect a larger than actual number of packets.

5.1.7.2.19 OMPS Telemetry

The OMPS Telemetry RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.20 OMPS LP Science

There are several elements that must be accounted for in percent complete calculations. The RDR maintains two databases, one for each APID. For each scan, it recalculates the total number of expected packets. The sum of the expected from each database becomes the total number expected. These RDRs are only created when the satellite is on the day side of the orbit.

5.1.7.2.20.1 Short Granule

The RDR must account for short granules. As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans, effectively changing the number of expected APs.

5.1.7.2.20.2 Super-Segmented Packet

Since scans can be configured on-board to be super-segmented, the RDR is sized to handle the scan possible. When the RDR receives the first packet of a super-segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. When the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING can now calculate the exact number of APs it should expect for that scan. Until both of these pieces of information is received, the RDR will expect a larger than actual number of packets for that scan.

5.1.7.2.21 OMPS LP Calibration

The Calibration RDR is an event based RDR. The event is made up of a configurable number of images. The RDR maintains a database for each image and recalculates the total number of expected after each database update. Additionally, special code enforces that all images of a specific event will be stored in a single RDR. These RDRs are only created periodically when the satellite is on the night side of the orbit.

5.1.7.2.21.1 Super-Segmented Packet

Each image is made up of a super-segmented group of packets (although that may mean a single segment or even a single packet). For each image, as the RDR receives the first packet of the segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. After the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING now knows the exact number of APs it should expect for that image. Until either of these pieces of information is received, the RDR must continue to expect the maximum number of packets for an image.

5.1.7.2.22 OMPS LP Diagnostic Exposure One

This RDR is sized to provide extensive flexibility for Cal/Val. It is sized to for the smallest temporal granule size allowable and the largest image (super-segmented) that Cal/Val could request. These RDRs are only created when the sensor is commanded into a diagnostic mode.

5.1.7.2.22.1 Super-Segmented Packet

Since images can be configured on-board to be super-segmented, the RDR is sized to handle the largest image possible. When the RDR receives the first packet of a super-segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. When the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING can now calculate the exact number of APs it should expect for that image. Until both of these pieces of information is received, the RDR will expect a larger than actual number of packets.

5.1.7.2.23 OMPS LP Diagnostic Exposure Two

This RDR is sized to provide extensive flexibility for Cal/Val. It is sized to for the smallest temporal granule size allowable and the largest image (super-segmented) that Cal/Val could request. These RDRs are only created when the sensor is commanded into diagnostic mode.

5.1.7.2.23.1 Super-Segmented Packet

Since images can be configured on-board to be super-segmented, the RDR is sized to handle the largest image possible. When the RDR receives the first packet of a super-segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. When the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING can now calculate the exact number of APs it should expect for that image. Until both of these pieces of information is received, the RDR will expect a larger than actual number of packets.

5.1.7.2.24 OMPS LP Diagnostic Calibration

This RDR is sized to provide extensive flexibility for Cal/Val. It is sized to for the smallest temporal granule size allowable and the largest image (super-segmented) that Cal/Val could request. These RDRs are only created when the sensor is commanded into diagnostic mode.

5.1.7.2.24.1 Super-Segmented Packet

Each image is made up of a super-segmented group of packets (although that may mean a single segment or even a single packet). For each image, as the RDR receives the first packet of the segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. After the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING now knows the exact number of APs it should expect for that image. Until either of these pieces of information is received, the RDR must continue to expect the maximum number of packets for an image.

5.1.7.2.25 OMPS NP Science

There are several elements that must be accounted for in percent complete calculations. The RDR maintains a database for each scan and recalculates the total number of expected after each database update. These RDRs are only created when the satellite is on the day side of the orbit.

5.1.7.2.25.1 Short Granule

The RDR must account for short granules. As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans, effectively changing the number of expected APs.

5.1.7.2.25.2 Super-Segmented Packet

Since scans can be configured on-board to be super-segmented, the RDR is sized to handle the scan possible. When the RDR receives the first packet of a super-segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. When the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING can now calculate the exact number of APs it should expect for that scan. Until both of these pieces of information is received, the RDR will expect a larger than actual number of packets.

5.1.7.2.26 OMPS NP Calibration

The Calibration RDR is an event based RDR. The event is made up of a configurable number of images. The RDR maintains a database for each image and recalculates the total number of expected after each database update. Additionally, special code enforces that all images of a specific event will be stored in a single RDR. These RDRs are only created periodically when the satellite is on the night side of the orbit.

5.1.7.2.26.1 Super-Segmented Packet

Each image is made up of a super-segmented group of packets (although that may mean a single segment or even a single packet). For each image, as the RDR receives the first packet of the segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. After the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING now knows the exact number of APs it should expect for that image. Until either of these pieces of information is received, the RDR must continue to expect the maximum number of packets for an image.

5.1.7.2.27 OMPS NP Diagnostic Earth View

This RDR is sized to provide extensive flexibility for Cal/Val. It is sized to for the smallest temporal granule size allowable and the largest image (super-segmented) that Cal/Val could request. These RDRs are only created when the sensor is commanded into diagnostic mode.

5.1.7.2.27.1 Super-Segmented Packet

Since images can be configured on-board to be super-segmented, the RDR is sized to handle the largest possible. When the RDR receives the first packet of a super-segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. When the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING can now calculate the exact number of APs it should expect for that image. Until both of these pieces of information is received, the RDR will expect a larger than actual number of packets.

5.1.7.2.28 OMPS NP Diagnostic Calibration

This RDR is sized to provide extensive flexibility for Cal/Val. It is sized to for the smallest temporal granule size allowable and the largest image (super-segmented) that Cal/Val could request. These RDRs are only created when the sensor is commanded into diagnostic mode.

5.1.7.2.28.1 Super-Segmented Packet

Each image is made up of a super-segmented group of packets (although that may mean a single segment or even a single packet). For each image, as the RDR receives the first packet of the segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. After the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING now knows the exact number of APs it should expect for that image. Until either of these pieces of information is received, the RDR must continue to expect the maximum number of packets for an image.

5.1.7.2.29 OMPS TC Science

There are several elements that must be accounted for in percent complete calculations. The RDR maintains a database for each scan and recalculates the total number of expected after each database update. These RDRs are only created when the satellite is on the day side of the orbit.

5.1.7.2.29.1 Short Granule

The RDR must account for short granules. As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans, effectively changing the number of expected APs.

5.1.7.2.29.2 Super-Segmented Packet

Since scans can be configured on-board to be super-segmented, the RDR is sized to handle the scan possible. When the RDR receives the first packet of a super-segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. When the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING can now calculate the exact number of APs it should expect for that scan. Until both of these pieces of information is received, the RDR will expect a larger than actual number of packets.

5.1.7.2.30 OMPS TC Calibration

The Calibration RDR is an event based RDR. The event is made up of a configurable number of images. The RDR maintains a database for each image and recalculates the total number of expected after each database update. Additionally, special code enforces that all images of a specific event will be stored in a single RDR. These RDRs are only created periodically when the satellite is on the night side of the orbit.

5.1.7.2.30.1 Super-Segmented Packet

Each image is made up of a super-segmented group of packets (although that may mean a single segment or even a single packet). For each image, as the RDR receives the first packet of the segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. After the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING now knows the exact number of APs it should expect for that image. Until either of these pieces of information is received, the RDR must continue to expect the maximum number of packets for an image.

5.1.7.2.31 OMPS TC Diagnostic Earth View

This RDR is sized to provide extensive flexibility for Cal/Val. It is sized to for the smallest temporal granule size allowable and the largest image (super-segmented) that Cal/Val could request. These RDRs are only created when the sensor is commanded into a diagnostic mode.

5.1.7.2.31.1 Super-Segmented Packet

Since scans can be configured on-board to be super-segmented, the RDR is sized to handle the scan possible. When the RDR receives the first packet of a super-segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. When the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING can now calculate the exact number of APs it should expect for that scan. Until both of these pieces of information is received, the RDR will expect a larger than actual number of packets.

5.1.7.2.32 OMPS TC Diagnostic Calibration

This RDR is sized to provide extensive flexibility for Cal/Val. It is sized to for the smallest temporal granule size allowable and the largest image (super-segmented) that Cal/Val could request. These RDRs are only created when the sensor is commanded into a diagnostic mode.

5.1.7.2.32.1 Super-Segmented Packet

Each image is made up of a super-segmented group of packets (although that may mean a single segment or even a single packet). For each image, as the RDR receives the first packet of the segment, it determines the exact number of segments to expect by extracting it from the OMPS Header of the packet. After the RDR receives the last segment of the group, it can determine the number of packets for it. As all previous segments have 256 packets, ING now knows the exact number of APs it should expect for that image. Until either of these pieces of information is received, the RDR must continue to expect the maximum number of packets for an image.

5.1.7.2.33 Spacecraft Diary

Spacecraft Diary RDRs does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations. This is true for both NPP and GCOM RDRs.

5.1.7.2.33.1 Fattened Granule

Due to there being several APIDs of different frequencies, a simple short granule implementation could not be used. This RDR is modified to size the RDR for an additional packet per APID in order to ensure there is enough storage for the variability.

5.1.7.2.34 Spacecraft Telemetry

The Spacecraft Telemetry RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations. This is true for both NPP and GCOM RDRs.

5.1.7.2.34.1 Fattened Granule

Due to the non-scan based nature of the data and the lack of temporal uniformity in the data, this RDR is fattened. This means that these RDRs will have to always time out and use thresholds to determine how to release.

5.1.7.2.34.2 Asynchronous Packets

This RDR collects numerous APIDs, many of which do not come all the time or must be requested to show up at all. As such, it is expected that the RDR never reaches 100% and is therefore released using the Timeouts and Thresholds.

5.1.7.2.35 VIIRS Science

VIIRS Science RDRs account for several dynamic components to calculate the expected number of APs. They are short granules, scan mode, and some additional commanded packets.

5.1.7.2.35.1 Short Granule

As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans. If it does not determine, the RDR is a short granule, it continues to expect N scans.

5.1.7.2.35.2 Day/Night

During operational science processing, when over a day portion of the earth, the instrument sends down all channels while over a night portion of earth, only a subset of the channels come down. This means that for each scan, the number of packets can vary due to the mode. For each new scan time, the database also tracks the mode (day vs. night) of the scan. Based on the number of expected scans and the mode of each of them, a new expected number of packets is calculated after each database update.

5.1.7.2.35.3 Asynchronous packets

Additionally, the VIIRS sensor can send down two additional APIDs during the night mode if commanded to. These two APIDs are not normally expected, but if they show up, they are added to the expected number of packets to balance the received.

5.1.7.2.36 VIIRS Diagnostic

VIIRS Diagnostic RDRs account for several dynamic components to calculate the expected number of APs. They are short granules and some additional commanded packets. Unlike the Science stream, the Diagnostic stream does not vary between night and day modes.

5.1.7.2.36.1 Short Granule

As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans. If it does not determine, the RDR is a short granule, it continues to expect N scans.

5.1.7.2.36.2 Asynchronous packets

Additionally, the VIIRS sensor can send down two additional APIDs during the night mode if commanded to. These two APIDs are not normally expected, but if they show up, they are added to the expected number of packets to balance the received.

5.1.7.2.37 VIIRS Memory Dump

The VIIRS Memory Dump RDR is designed to handle the dynamic nature of memory dumps. Since the VIIRS instrument fulfills all memory dumps as a segmented packet, the RDR is sized to handle the largest byte request possible. When the RDR receives the first packet of the segment, it determines the exact number of packets to expect by extracting it from the secondary header of the packet.

5.1.7.2.37.1 Asynchronous Packet

Memory Dump packets are a commanded data. They do not show up regularly. When a request is made of the satellite, it responds by sending down a memory dump stream as a response.

5.1.7.2.38 VIIRS Telemetry

The VIIRS Telemetry RDR must account for short granules in its percent complete calculations..

5.1.7.2.38.1 Short Granule

As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans. If it does not determine, the RDR is a short granule, it continues to expect N scans.

5.1.7.2.39 VIIRS Diagnostic Telemetry

The VIIRS Diagnostic Telemetry RDR must account for short granules in its percent complete calculations..

5.1.7.2.39.1 Short Granule

As each new packet is received, the RDR updates its internal database of scan times. Once the RDR has received (N-1) scans of its expected N, it does an analysis of the timestamps to see if there are any gaps that would imply another scan is coming. If it does not find a gap, it assumes the RDR only receives (N-1) scans. If it does not determine, the RDR is a short granule, it continues to expect N scans.

5.1.7.2.40 AMSR2 Science

The AMSR2 Science RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.40.1 Fattened Granule

Due to the lack of sensor information defining error margins, etc, variability could not be determined for short granule analysis. This means that these RDRs will have to always time out and use thresholds to determine how to release.

5.1.7.2.41 AMSR2 Telemetry

The AMSR2 Telemetry RDR does not have a dynamic component to its percent complete calculations. The initial expected number of APs is used for all calculations.

5.1.7.2.41.1 Fattened Granule

Due to the lack of sensor information defining error margins, etc, variability could not be determined for short granule analysis. This means that these RDRs will have to always time out and use thresholds to determine how to release.

5.2 MSD

5.2.1 Startup parameters

The same MSD executable is used for both the dynamic processing of MSD and for software updates. The two modes are distinguished by a startup parameter. The normal configuration for the process is persistent mode and does not require any startup parameters. To use transient mode, also known as software update mode, an argument of "-transient" is added to the command line upon executable startup. Ignoring the startup mechanism (light-weight vs. INF), start up of the transient mode of ING MSD would look like the following:
IngMsd.exe –transient

5.2.2 File Naming conventions

MSD file naming conventions convey information about a specific MSD file that is needed for downstream processing. These include things like start and end effectivity, product origin, satellite name, and so on. These file names are then parsed by Ingest and the various components of the name are placed into metadata, and occasionally as part of the internal format, as needed. The naming formats for MSD are described in the DPIS ICD, Volume 1.

5.2.3 Landing Zones

Landing zones are integral to the proper functioning of ING MSD. At its core, a landing zone is simply a directory where MSD of a certain type is placed so that ING MSD can process it. They are intended to provide a partitioning of MSD products according to either product type or product source. Each landing zone has two attributes associated with it. The first attribute is a file completion strategy and the second is a file naming format.

The file completion strategy of a landing zone determines what criteria the ING MSD process uses to determine if a file in that landing zone is complete. The file naming format indicates the naming convention an incoming MSD file uses to convey pertinent information regarding that MSD. This information includes things such as the data provider, effective lifespan of the data, and forecast time, though the exact contents vary depending on the type of MSD being represented. This is covered more in depth in the sections specifically covering file naming formats and file completion strategies.

A file placed into a landing zone with a mismatched file naming format is rejected. An example of this is an ancillary file placed in an auxiliary landing zone. Similarly, if a file is placed into a landing zone with an improper completion strategy it may be rejected. However, this is not guaranteed to occur and the handling of an improper completion strategy on a file depends on the specific file completion strategy involved.

Additionally, landing zones are partitioned based on the process type at the most basic level. Both the persistent MSD process and the transient MSD process monitor a set of discrete

landing zones. By default, there is no overlap in landing zone coverage between the two processes.

Although not actually a landing zone location, an additional directory is found in the directory structure and is used by both modes. This directory is known as the Temp directory and is a temporary directory that ING uses to move unusable files for a short period of time. After a period of time each file is purged from this location as well. For the purposes of ING MSD, unusable simply means that the file could not be processed by ING MSD at this time either due to a configuration error or some problem with the data itself.

5.2.3.1 Persistent Mode

For the persistent mode, the following landing zones are currently monitored for MSD:

- ODAD -This directory receives all the Official Dynamic Ancillary Data (ODAD) files. This includes NCEP, NAAPS, NOGAPS, and Earth Orientation data. All such files are received using the ancillary file naming convention.
- SDAD - This directory receives all the Substitute Dynamic Ancillary Data (SDAD) files. There is no specific data for SDAD: the centrals have the ability to substitute data of their choice for any ODAD. The only constraints is that SDAD files should be in the same format and naming convention as ODAD files, including Collection Short Name, as outlined in CDFCB-X Volume 6, D34862-06. All such files are received using the ancillary file naming convention.
- C3S - This directory receives all the C3S products that are brought into IDPS. This includes Two Line Element (TLE), Revolution Table, Spacecraft Configuration Database Update, Manually Initiated Processing Coefficients and Mission Schedule data. All such files are received in the auxiliary file naming convention.
- DQM - This directory receives all the DQM products that are brought into IDPS. This includes Data Quality Threshold Tables and Processing Coefficients data. All such files are received in the auxiliary file naming convention.
- GCOM_TLE – This directory receives all the GCOM TLE files that are sent to IDPS from KSAT

5.2.3.2 Transient Mode

In transient mode, the landing zones being polled are currently configured to the following:

- StaticAnc - This directory receives all the Static MSD files that are not tiled, using the ancillary file naming convention.
- StaticTile - This directory receives all the static products that use the tiled naming conventions. This includes files such as Terrain Ecology (Tereco).
- LUT - This directory receives the look up table products using the auxiliary file naming convention.
- SeededIP - This directory receives the seeded IP files in the tiled naming convention.

5.2.4 File Completion Strategies

File completion strategies dictate when a given MSD file is viewed as ready for processing by the ING MSD process after being placed in its appropriate landing zone. Generally this means that incoming files possess two distinct naming states, one signifying a file transfer in process and one signifying the transfer is complete. Currently there are two file completion strategies, the suffix completion strategy and the checksum completion strategy.

Regardless of the strategy being used, if an MSD file is left in its interim format for too long, it is removed from the landing zone and placed in the "Temp" holding directory so the operator can examine it.

5.2.4.1 Suffix Completion

The suffix completion strategy relies on a file being appended with a configurable extension while it's in transit and then that extension being removed when the file is completely transferred over. The configuration guide parameters governing this are described in the ING SI SW Configuration Guide User Manual. Currently the in-transit suffix is configured to be .tmp. Therefore to transfer a file successfully to an Ingest landing zone you would first copy the file over as ***file name.*tmp** and then rename it to ***file name.suffix*,** where ***file name*** is the name of the file being transferred and ***.suffix*** is the original suffix affixed to that file name. There is a list of approved suffixes associated with this strategy. If a file does not contain a known suffix from that list, it is rejected as invalid.

5.2.4.2 Checksum Completion

The checksum completion strategy relies on both an MSD file and its accompanying checksum file being copied over in an interim format and then renamed to their proper names. The checksum completion strategy is not configurable. Currently, while data files are in transit, they are expected to use a suffix of TMPFILENAME. While the accompanying checksum files are in transit, they are expected to use a suffix of TMPCMP_FILENAME. Once the files are present in the landing zone, data files are to be renamed to use their original suffix and the checksum files corresponding to that file should be renamed to have a suffix of CMP_FILENAME. The values used for the various checksum suffixes are configurable and as such are detailed in the ING SI SW Configuration Guide User Manual.

5.2.5 Polling of landing zones

Upon startup, ING MSD immediately polls all of its landing zones for MSD, starting with the first listed landing zone in the configuration guide and ending with the last. After the initial poll, subsequent polls are conducted in a periodic fashion. When no data is found on any of the Landing Zones, the process sleeps for a configurable period defined by the ING.MSD.NO_DATA_SLEEP_INTERVAL attribute of the configuration guide, as described in the ING SI SW Configuration Guide User Manual.

When a directory is polled, the ING MSD process examines each file contained in a landing zone and attempts to match it against the expected file naming convention and file completion strategy being used, as described in sections 5.2.2 and 5.2.4. Any non-conforming files are sent to the temporary holding directory.

In transient mode, ING MSD polls each Landing Zone once, and shuts down.

5.2.6 Purge of older versions

MSD can be configured to purge older MSD. For the MSD process, purging means that only the most recent file(s) of a given MSD type are kept in DMS after an insert of new data. However not every MSD type is purged by ING and the number of files retained is configurable. Both of these variables are controlled via the configuration guide and can be found in the ING SI SW Configuration Guide User Manual.

The configuration of purging is controlled at the product type level, not on a short name level. This means that all processing coefficients for instance share the same configuration settings. Versioning is performed after a file has been ingested by searching for any entries in DMS that match the file ingested. Generally the search criteria consists of just the collection short name, but occasionally other attributes like Tile ID are used to help obtain an accurate list of matches. Once a list of matches is assembled, MSD deletes all but the most recent N entries, where N is defined in the configuration guide as described above. This ensures that DMS doesn't become filled with outdated or irrelevant MSD entries.

The two configuration guide entries that control this behavior are ING.MSD.<product name>.WILL_PURGE_DMS and ING.MSD.<product name>.NUMBER_OF_DMS_COPIES. The product name can be any of the following product type identifiers; Coefficients, NOGAPS, NCEP, NAAPS, Static, SeededIP, LUT, Revolution Table, Earth Orientation, Mission Schedule, SCDBU, TLE, and Data Quality Thresholds. The WILL_PURGE_DMS attribute is a value that can be set to either true or false and indicates whether the ING process is responsible for purging DMS of outdated files. The NUMBER_OF_DMS_COPIES indicates how many copies of a given product are kept in DMS, if versioning is enabled. Additional detail can be found in the ING Configuration Guide User Manual.

5.2.7 Preventing older versions

ING MSD also has functionality to prevent the processing of data that is older than data already contained in DMS. This means that if an MSD file exists in DMS, and an older file (based on the effectivity of the two files) is dropped into an ING landing zone, the older file is rejected by ING and a message is sent to the log indicating the rejection. Normally the check for older data is fairly simple and overlap is determined by comparing the short name and effectivity of incoming MSD to that of MSD already in DMS. If the incoming DMS has a short name that is the same as a file already in DMS and that file in DMS has a starting effectivity

that is later than the start effectivity of the incoming MSD, the incoming MSD is rejected by ING MSD. This guarantees that older data is not introduced into the system.

However, the checks for forecast data are more complicated due to the fact that by design, forecast datasets have multiple files mapped to the same short name. So ingested a forecast file with a lesser start effectivity isn't necessarily an error. For example, if ING receives a dataset with a missing/erroneous 6 hour forecast, and we try to process that missing 6 hour forecast at a later time, the simple logic used for other MSD will cause that file to be rejected since it has a shortname that matches other products in DMS, and the later forecasts will have a start effectivity greater than that of the incoming file. So, to accommodate forecast data, ING MSD has special logic to determine whether they should be rejected. When receiving a forecast file, the process will query on both the short name of the file and the effectivity. It will then reject the incoming forecast file if and only if its synoptic time is also less than any existing synoptic times that match its initial short name and effectivity query.

5.2.8 Location of erroneous MSD

When a file does not conform to the expected format, either through an erroneous file name, improper completion strategy or corrupt data, it is placed in a separate directory to be examined by the operator or other personnel. The files placed there remain intact for a configurable amount of time. Once that time period has elapsed, the file is removed. The actual time period before a purge take places is defined within the ING.MSD. CLEANUP_PERIOD attribute of the configuration guide, as described in the ING SI SW Configuration Guide User Manual. The location of this directory is defined by the ING.MSD.TRASH_LOCATION attribute. Review the ING Configuration Guide User Manual for additional detail.

5.2.9 Effectivity

Unlike most of the other types of data in IDPS, MSD cannot be identified using a granule ID. Instead, MSD is given an effectivity range. Every file naming convention has both a start effectivity field and an end effectivity field of some sort. These fields contain a timestamp, usually in the rough form of YYYYMMDDHHMM. The two timestamps denote when the MSD is actually valid, giving it a discrete start and expiration date. If the end timestamp in a file is set to all 0's, then the MSD is said to have unbounded effectivity, which means that it is valid until the end of the IDPS epoch.

5.2.10 Forecast data

Forecast data is handled a bit differently by the MSD process. The various ways in which it differs are described below.

5.2.10.1 Effectivity

Forecast data is a special case when it comes to effectivity. Rather than assigning the effectivity range from a start and end effectivity field in the file name, ING calculates the effectivity range from the synoptic and forecast time in the filename and an effectivity window size from a configuration guide. The synoptic time of forecast data is basically the 0 hour of the model run. The forecast time is given in hours, and indicates the time past the synoptic time that the forecast is projecting data for.

The window size indicates how big the effectivity period should be for a given forecast. So if the window size was set to an hour, the difference between the start effectivity time and the end effectivity time would be an hour as well. Start effectivity is then computed by adding the forecast time to the synoptic time and then subtracting half of the appropriate effectivity window. End effectivity is computed by again adding the forecast time to the synoptic time and then adding half of the appropriate effectivity window.

This window size is defined within the ANC.INTERNAL_FORECAST_WINDOW and ANC.NATIVE_FORECAST_WINDOW fields as described in the ING SI SW Configuration Guide User Manual. The INTERNAL_FORECAST_WINDOW field indicates the window sizing to be used for internally formatted forecast products, whereas the NATIVE_FORECAST_WINDOW field indicates the window size for native ones.

5.2.10.2 Minimum time until use

The EDR-IR specifies a minimum time that must elapse before certain ancillary data is to be used after being produced. This manifests itself in IDPS through a manipulation of the start effectivity of certain ancillary products with a given forecast. The minimum time until a product can be used is then defined as the lowest acceptable start effectivity a piece of MSD of a given type can have. This minimum time is calculated by taking a configurable value from the configuration guide, representing the time specified in the EDR-IR, and adding it to the synoptic time of a file. This time then represents how far past the time of generation (the synoptic time) a file must be, before it is considered valid and thus can be used. The configurable value is contained in the ANC.<product name>.TIME_UNTIL_USE field described in the ING SI SW Configuration Guide User Manual.

If, using the calculations for effectivity in section 5.2.10.1, a file has a start effectivity that is earlier than the minimum time until use, the MSD process alters the start effectivity so that it instead equals the minimum time until use. This ensures that a given file won't be used until it's appropriate.

5.2.10.3 Extended forecasts

Some forecasts have their effectivity extended in order to provide extended coverage in the event of a prolonged period of missing data. For NCEP, NAAPS, and NOGAPS, the 24 hour

forecast is extended. For Earth Orientation data, all forecasts are considered extended. When a forecast is extended, its end effectivity is set to a much larger value than it otherwise would be. The actual extent of the increase is defined in the ANC.<product name>.24HR_FORECAST_EXTENSION_PERIOD field located in the ING SI SW Configuration Guide User Manual. That field describes how much a forecast is extended by, when a forecast is an extended forecast

5.2.10.4 Forecast Based Collection Short Name variations

For the NCEP, NOGAPS and NAAPS datasets there can be varying collection short names depending on the forecast in question. Normally the collection short name of these three products' internally formatted data is of the form <product>-ANC-Int, where product is the data type in question. However, when temporal interpolation is required, the short name of the 3 hour forecast differs to prevent the tasking of three hour forecasts by Infrastructure. In these cases the collection short name of three hour forecasts follows the form of <product>-03HR-ANC-Int by default.

The actual collection short names used for three hour forecasts are configurable, however. The field used to configure these entries is found in the ANC_CFG_Shared.xml. The actual field depends on the product and is found in ANC.<product>.THREE_HOUR_INTERNAL_SHORT_NAME, as shown in Table 3.3.2.5 of the ING SI SW Configuration Guide User Manual.

5.2.10.5 Substitute Data Based Collection Short Names

For the NCEP, NOGAPS and NAAPS datasets there can be varying collection short names depending on the state of the MSD in question as well. In addition to the nominal Official Dynamic Ancillary Data (ODAD) format, Dynamic Ancillary Data can also be brought into the system as Substitute data. ING MSD has special collection short name rules for substitute data, to distinguish them from ODAD. The collection short name of Substitute Dynamic Ancillary Data (SDAD) always has –SUB appended to the end. So, say the collection short name of an internal 6 hour NCEP forecast would be NCEP-ANC-Int if it was brought in as ODAD. If that same file were brought in as SDAD, its collection short name would instead be NCEP-ANC-Int-SUB.

5.2.10.6 Temporal Interpolation

The forecast products NCEP, NAAPS and NOGAPS are temporally interpolated by the related PRO process after release by ING MSD. To support temporal interpolation, the elements described above in sections 5.2.10.1 through 5.2.10.4 are present within ING. The effectivity of the forecast data is mutable to allow space for temporally interpolated files to be inserted to create a continuous flow of data. The collection short names of three hour forecasts are different to allow for the appropriate tasking of temporal interpolation.

If temporal interpolation is disabled, the following changes must take place for ING to support the lack of interpolation. First, the ANC.INTERNAL_FORECAST_WINDOW field should be set to the same value as the ANC.NATIVE_FORECAST_WINDOW. This ensures that the data remains continuous in the absence of temporally interpolated forecasts. Second, the ANC.<product>.THREE_HOUR_INTERNAL_SHORT_NAME field should be reset to use the nominal collection short name conventions for the product in question. The convention is currently <product>-ANC-Int, where <product> is the name of the product in question (NCEP, NOGAPS, or NAAPS).

5.2.11 Configuring landing zone monitoring

The ability to determine which landing zones are being monitored by a given MSD process is completely configurable and can be modified as described below. The relevant configuration guide entries can be found in the IDPS ING Software Configuration Guide User Manual.

If one is altering the landing zones being monitored, the first thing to examine is the identifier's landing zones are being labeled with. This list of identifiers is found under the ING.MSD.PROCESS_TYPE.TRANSIENT.DIRECTORY_MONITOR_LIST and ING.MSD.PROCESS_TYPE.PERSISTENT.DIRECTORY_MONITOR_LIST configuration guide entries. Under each of those entries, a colon delimited list of values is present, such as LZ5:LZ6:LZ7:LZ8 as an example. Each entry in that list is a mnemonic for a given landing zone. An example is listed below:

```
<group name="ING">
   <group name="MSD">
    …
      <!--
       Group Name:  Process Type
       Description:
          This group describes the landing zones monitored by each process
          type
      -->
      <group name="PROCESS_TYPE">

         <!--
          Group Name:  Transient
          Description:
             This group describes the landing zones monitored by transient
             processes
         -->
         <group name="TRANSIENT">
            <!--
             A colon delimited list of landing zones monitored by this
             process
            -->
            <config>
               <name>DIRECTORY_MONITOR_LIST</name>
               <configValue>LZ5:LZ6:LZ7:LZ8</configValue>
            </config>
```

```
   </group>

   <!--
    Group Name:  Persistent
    Description:
       This group describes the landing zones monitored by
       persistent processes
   -->
   <group name="PERSISTENT">
      <!--
       A colon delimited list of landing zones monitored by this
       process
      -->
      <config>
         <name>DIRECTORY_MONITOR_LIST</name>
         <configValue>LZ1:LZ2:LZ3:LZ4:LZ9</configValue>
      </config>
   </group>
</group>
```

These mnemonics then map to a different set of configuration guide entries that take the form of ING.MSD.DIRECTORY_MONITOR.<landing zone name>, where landing zone name is the mnemonic assigned to a given landing zone. An example extending the previous one is contained below:

```
<group name="ING">
   <group name="MSD">
    …
      <!--
       Group Name:  Directory Monitor
       Description:
         This group describes the format of the various landing
         zones monitored by ING MSD.  Each entry below is
         identified by the name of the landing zone and then its
         contents.  Each landing zone has the following entries;

         * Directory:  The directory that this landing zone
           represents
         * LZ Type:  The type of MSD that is being placed in this
           landing zone.  This determines the expected file
           naming format.
         * Completion Strategy: The completion strategy being used
           to determine if a given MSD file is complete, or
           still in transit.
      -->
      <group name="DIRECTORY_MONITOR">
         <group name="LZ1">
            <config>
               <name>DIRECTORY</name>
               <configValue>$(ING.MSD.LZ_ROOT)/ODAD</configValue>
            </config>
            <config>
               <name>LZ_TYPE</name>
               <configValue>ANC</configValue>
```

```
      </config>
      <config>
         <name>COMPLETION_STRATEGY</name>
         <configValue>CHECKSUM</configValue>
      </config>
   </group>
```

In this example, one of the landing zones is defined. The first part of this definition is the directory name, “LZ1” in this case. The rest of the example concerns the specific attributes of a landing zone, all of which are configurable. The DIRECTORY field is used to denote the physical location of the landing zone, which can be any directory readable by the ING executable. The LZ_TYPE field contains the landing zone type, which is a shorthand way of referring to what file naming format is expected for this landing zone. Valid values are ANC for ancillary data, AUX for auxiliary data and Tile for tiled datasets. Finally, the COMPLETION_STRATEGY field indicates what file completion strategy is being used to determine whether or not a given file is complete. Valid values for this field are either CHECKSUM or SUFFIX.

Manipulation of existing landing zones is fairly straightforward. Any of the sub-fields of a landing zone can be manipulated without substantial risk. If an entry is edited to be outside the acceptable range, for instance a completion strategy is set to something other than checksum or suffix, the ING process terminates and an error is reported in the log.

Creation of new landing zones and/or the radical reconfiguration of landing zones is also possible, but carries more substantial risk. To add a new landing zone, first you need to devise an identifier for it, similar to the ‘LZ1’ identifier being used above. Once that is done, the identifier must be added to the appropriate listing for whichever process type the landing zone is being associated with. So, for instance to add ‘LZ9’ to the transient MSD process you would modify the ING.MSD.PROCESS_TYPE.TRANSIENT.DIRECTORY_MONITOR_LIST field to contain ‘LZ5:LZ6:LZ7:LZ8:LZ9’.

Once this is done, the next step is to derive a new set of XML fields corresponding to the new landing zone. The XML must all go under the DIRECTORY_MONITOR group. The easiest way to do this is to simply copy an existing landing zone and modify the appropriate fields. .

6.0 MESSAGES AND ERRORS

6.1 ERROR RECOVERY

6.1.1 SMD

There are four varieties of errors in SMD. The first type is induced by a failure of the operating system, DMS, or INF. These failures are considered catastrophic and ING SMD terminates after encountering one. In order to recover from these errors the underlying cause for the failure must be discovered and rectified before restarting the Ingest process.

The second type of error is a configuration error within Ingest SMD. This too is a fatal error, and results in process termination. In this case the improper configuration entry needs to be fixed before ingest can be restarted.

The third class of errors is errors involving the connection to the data provider. These errors can be remedied by checking to ensure that the data provider and ingest are using the same DNS keys and that the data provider is giving ingest a valid port and server to connect to. ING SMD continues to log this error until the process is shutdown or the issue is resolved.

The final category of errors consists of those that are within a data stream. This includes things such as malformed packets, invalid APIDs, and other things affecting the stream of stored mission data coming in from the DHN. If the problems are minor and sporadic they are simply ignored by ING SMD. If the number of consecutive errors exceeds a configurable threshold, however, the process disconnects the faulty data stream and attempts to reconnect to the data provider.

6.1.2 MSD

There are four varieties of errors in MSD. The first type is induced by a failure of the operating system, DMS, or INF. These failures are considered catastrophic and Ingest MSD terminates upon encountering one. In order to recover from these errors the underlying cause for the failure must be discovered and rectified before restarting the Ingest process.

The second type of error is a configuration error within ING MSD. This too is a fatal error, and results in process termination. In this case the improper configuration entry needs to be fixed before ING can be restarted.

The next class of errors is improperly configured MSD files. In this event, the file is rejected by the MSD process and moved to a separate directory where the operator can examine it to see why it was rejected. In this case no error recovery is necessary, as the process does not terminate, but simply continues processing other files.

The final set of errors are those that occur when a file is configured correctly, meaning it uses the appropriate file naming convention and completion strategy, but the actual data is corrupt or incorrect. When an error of this type occurs, the file is rejected by the ING MSD process and moved to a separate directory for later analysis. A message is sent to the log indicating that an error occurred. In this case, the ING process does not terminate and attempts to process other files.

6.2 MESSAGES

6.2.1 Status/Details

The Status and Details message interfaces are defined in the DPIS ICD Vol 1. These provide the standard messages that are displayed to the operator GUI including process and data status. Message content and possible operator actions are documented in Appendix A.

6.2.2 Performance

Performance messages are special messages that take a snapshot of the system and provide timestamps for specific operations (e.g. start and end of an RDR) to determine performance of ING functionality. These are stored in a separate file(s) per process when enabled on the IDP GUI. The performance metrics required for the ING Subsystem are defined in the DPIS ICD Vol 1, but there will be custom messages used for ING development in the files as well.

6.2.3 Debug

Debug messages provide low level information to a developer to determine the cause of a problem and the state of the SW at the time of a error. Debug messages are defined as “at developer’s discretion”, so they can be used for many circumstances. There are various levels of debug to allow filtering for just errors and various levels of fidelity. The levels are defined in the DPIS ICD Vol 1. Debug messages are stored in a separate file(s) when enabled on the IDP GUI.

6.3 STATUS/DETAILS MESSAGE CONTENT

6.3.1 Common

6.3.1.1 PROC_INIT_COMPL

PROC_INIT_COMPL	
Message Description	Indicates that the ING process has completed its initial startup routine and is beginning normal operations.
Content	SMD Processes – "*satellite* :*configuration*:Initialization Complete" MSD Processes – MSD Master has been initialized
Variable Data	satellite - Satellite whose data is being read by this SMD instance as indicated by the –ingSC flag on SMD startup configuration - Configuration of the SMD instance as indicated using the –ingPROC flag on SMD startup
Recommended Action	No action needed

6.3.1.2 ING_ERR

ING_ERR	
Message Description	This message indicates that the ING process has encountered some sort of error. If the error is non-fatal then processing continues. Otherwise the process terminates.
Content	SMD Processes – "*satellite* :*configuration*: free form text of the error" MSD Processes – "free form text of the error"
Variable Data	**satellite** - Satellite whose data is being read by this SMD instance as indicated by the –ingSC flag on SMD startup **configuration** - Configuration of the SMD instance as indicated using the –ingPROC flag on SMD startup
Recommended Action	Check for common error conditions according to the relevant standard operating procedures. Depending on the repeatability and frequency of the error, the debug log can be enabled to provide a detailed debug log.

6.3.1.3 ING_INFO

ING_INFO	
Message Description	This message is used for nominal conditions in either the SMD or MSD process which do not require user action, but should still be noted.
Content	SMD Processes – "*satellite* :*configuration*: free form text" MSD Processes – "free form text"
Variable Data	**satellite** - Satellite whose data is being read by this SMD instance as indicated by the –ingSC flag on SMD startup **configuration** - Configuration of the SMD instance as indicated using the –ingPROC flag on SMD startup
Recommended Action	No action needed

6.3.2 SMD Specific

6.3.2.1 ING_FREE_FORM

ING_FREE_FORM	
Message Description	Non-critical information
Content	*"satellite* :*configuration*: <informative message>"
Variable Data	satellite - Satellite whose data is being read by this SMD instance as indicated by the –ingSC flag on SMD startup configuration - Configuration of the SMD instance as indicated using the -ingPROC flag on SMD startup
Recommended Action	No action needed

6.3.2.2 DROP_PKT

DROP_PKT	
Message Description	Tracks the number of packets that ING SMD received but did not store into an RDR.
Content	*"satellite* :*configuration*: **DropReason:PktsDropped**: <informative text>"
Variable Data	**satellite** - Satellite whose data is being read by this SMD instance as indicated by the –ingSC flag on SMD startup **configuration** - Configuration of the SMD instance as indicated using the –ingPROC flag on SMD startup **DropReason** – Specific string representing a summary reason for the drop. Valid values include: 1) INVALID_C3S_HEADER – size or version in one of the C3S extended headers does not match expected 2) UNEXPECTED_APID – received an APID that is not assigned to an RDR or part of the assigned drop list 3) MISMATCHED_GROUP – an APID had a group flag that was unexpected based on ING configuration (e.g. continuation packet when expected standalone 4) INDETERMINATE_OBS_TIME – continuation or last packet received without the first packet that provides timestamp (broken segment) 5) CONFIGURED_DROP – expected to be dropped 6) RDR_PKT_TRACKER_FULL – RDR did not have enough slots for this APID. 7) RDR_STORAGE_FULL – RDR did not have enough storage space for this packet 8) DUPE_PKT – duplicate packet received 9) DUPE_DIFFERENT_SIZE – duplicate packet received with a different size than original 10) OBS_TIME_OUT_OF_RANGE – packet received that has a timestamp previous to the granule ID base time and therefore cannot be assigned to an RDR 11) INVALID_APID_TO_RDR – an attempt to store an APID not assigned to the particular RDR **PktsDropped** – The number of packets that this message represents as dropped. This is needed as some drop messages collect a summary count for a period of time to minimize (throttle) the number of messages.

Recommended Action	**INVALID_C3S_HEADER** – Coordinate with C3S. This could be related to corrupt data in streams **UNEXPECTED_APID** – Coordinate with C3S. This could be related to corrupt data in streams **MISMATCHED_GROUP** – Most likely due to corruption of data stream (bit flips). If happening consistently on the same APID, further analysis is necessary **INDETERMINATE_OBS_TIME** – Usually no action required. Normally due to a gap in data that causes missing first packet of a segment. C3S should send the data as a retransmit **CONFIGURED_DROP** – No action required. This message is for traceability only. **RDR_PKT_TRACKER_FULL** – Investigation required as APID is not coming at expected rate **RDR_STORAGE_FULL** – Investigation required as packets are taking more space than expected **DUPE_PKT** – No action required. Duplicates are expected from retransmits **DUPE_DIFFERENT_SIZE** – Investigate with C3S as this should not happen based on C3S fill methodology **OBS_TIME_OUT_OF_RANGE** – Coordinate with C3S that sensor was re-booted or something similar as this should only happen when internal clocks are reset causing timestamps of 1-1-1958. **INVALID_APID_TO_RDR** – This should never happen in Operations, but is there as a double check for development testing

6.3.2.3 RDR_START

RDR_START	
Message Description	This message indicates that the creation of an RDR has started.
Content	New RDR Version Started **or** RDR started
Variable Data	None
Recommended Action	No action required

6.3.2.4 RDR_FAIL

RDR_FAIL	
Message Description	Two conditions can cause this message. One occurs after an unrecoverable crash in Ingest, resulting in corrupt RDRs being present in DMS. These RDRs are retrieved on startup and determined to be corrupt. In this event, a failure message is generated and a retransmit request is performed for the corrupt data. The second condition is a problem accessing the DMS interface.
Content	"Removing corrupt RDR in DMS and requesting data be retransmitted" **or** "Update of RDR to DMS Unsuccessful"
Variable Data	None
Recommended Action	For a corrupt RDR, a retransmit request is initiated to recover the lost data. No additional action required. For a DMS error, ING will shutdown and further investigation is necessary to determine why the DMS interface is not working.

6.3.2.5 RDR_GRAN

RDR_GRAN	
Message Description	RDR is ready to be further processed. The RDR_GRAN message contains an XML string that is used for data accountability metrics by INF.
Content	"<RDRXML><RDRQuality>*percent complete*</RDRQuality> <DataAccountabilityMetrics><numOfAPs>*received packets* </numOfAPs> <numIncompleteAPs>*incomplete packets* </numIncompleteAPs> <numAPsRepaired>*repaired packets*</numAPsRepaired> <granStartBoundaryTime>*granule start time*</granStartBoundaryTime> </DataAccountabilityMetrics></RDRXML>"
Variable Data	*percent complete* - Percentage of the RDR that is filled *received packets* - Number of received packets *incomplete packets* - Number of incomplete packets *repaired packets* - Number of repaired packets *granule start time* - Start time of the granule in IET
Recommended Action	No action needed

6.3.2.6 RDR_PARTIAL

RDR_PARTIAL	
Message Description	RDR is not ready to be processed. The RDR_PARTIAL message contains an XML string that is used for data accountability metrics via INF.
Content	"<RDRXML><RDRQuality>*percent complete*</RDRQuality> <DataAccountabilityMetrics><numOfAPs>*received packets* </numOfAPs> <numIncompleteAPs>*incomplete packets* </numIncompleteAPs> <numAPsRepaired>*repaired packets*</numAPsRepaired> <granStartBoundaryTime>*granule start time*</granStartBoundaryTime> </DataAccountabilityMetrics></RDRXML>"
Variable Data	*percent complete* - Percentage of the RDR that is filled *received packets* - Number of received packets *incomplete packets* - Number of incomplete packets *repaired packets* - Number of repaired packets *granule start time* - Start time of the granule in IET
Recommended Action	No action needed

6.3.3 MSD Specific

6.3.3.1 ANC_RECV

ANC_RECV	
Message Description	This message indicates that the ancillary MSD from the file indicated above was processed successfully and any downstream processing using the file can commence.
Content	Ancillary product *file name* received
Variable Data	The file name of the product as indicated above
Recommended Action	No action needed

6.3.3.2 ANC_FAIL

ANC_FAIL	
Message Description	This message indicates that the ancillary MSD from the file indicated above was not processed due to some sort of error condition.
Content	Ancillary product *file name* failed
Variable Data	The file name of the product as indicated above
Recommended Action	Check for common error conditions according to the relevant standard operating procedure. Depending on the repeatability of the error, the debug log could be enabled to provide a detailed debug log.

6.3.3.3 MSNSCHED_RECV

MSNSCHED_RECV	
Message Description	This message indicates that the mission schedule MSD from the file indicated above was processed successfully and any downstream processing using the file can commence.
Content	Mission Schedule MSD product ***file name*** received
Variable Data	The file name of the product as indicated above
Recommended Action	No action needed

6.3.3.4 MSNSCHED_FAIL

MSNSCHED_FAIL	
Message Description	This message indicates that the mission schedule MSD from the file indicated above was not processed due to some sort of error condition.
Content	Mission Schedule MSD product ***file name*** failed
Variable Data	The file name of the product as indicated above
Recommended Action	Check for common error conditions according to the relevant standard operating procedure. Depending on the repeatability of the error, the debug log could be enabled to provide a detailed debug log.

6.3.3.5 SCDB_RECV

SCDB_RECV	
Message Description	This message indicates that the spacecraft configuration database MSD from the file indicated above was processed successfully and any downstream processing using the file can commence.
Content	SCDB MSD product *file name* received
Variable Data	The file name of the product as indicated above
Recommended Action	No action needed

6.3.3.6 SCDB_FAIL

SCDB_FAIL	
Message Description	This message indicates that the spacecraft configuration database MSD from the file indicated above was not processed due to some sort of error condition.
Content	SCDB MSD product *file name* failed
Variable Data	The file name of the product as indicated above
Recommended Action	Check for common error conditions according to the relevant standard operating procedure. Depending on the repeatability of the error, the debug log could be enabled to provide a detailed debug log.

6.3.3.7 MSD_RECV

MSD_RECV	
Message Description	This message indicates that the auxiliary MSD from the file indicated above was processed successfully and any downstream processing using the file can commence. Despite the generic title, the MSD_RECV message is only used for auxiliary data currently, with ancillary data being covered by the ANC_RECV message.
Content	Auxiliary MSD product ***file name*** received
Variable Data	The file name of the product as indicated above
Recommended Action	No action needed

6.3.3.8 MSD_FAIL

MSD_FAIL	
Message Description	This message indicates that the MSD from the file indicated above was not processed due to some sort of error condition. Despite the generic title, the MSD_FAIL message is only used for auxiliary data currently, with ancillary data being covered by the ANC_FAIL message.
Content	Auxiliary MSD product ***file name*** failed
Variable Data	The file name of the product as indicated above
Recommended Action	Check for common error conditions according to the relevant standard operating procedure. Depending on the repeatability of the error, the debug log could be enabled to provide a detailed debug log.

6.3.3.9 STATIC_RECV

STATIC_RECV	
Message Description	This message indicates that the static MSD from the file indicated above was processed successfully and any downstream processing using the file can commence.
Content	Static MSD product ***file name*** received
Variable Data	The file name of the product as indicated above
Recommended Action	No action needed

6.3.3.10 STATIC_FAIL

STATIC_FAIL	
Message Description	This message indicates that the static MSD from the file indicated above was not processed due to some sort of error condition.
Content	Static MSD product ***file name*** failed
Variable Data	The file name of the product as indicated above
Recommended Action	Check for common error conditions according to the relevant standard operating procedure. Depending on the repeatability of the error, the debug log could be enabled to provide a detailed debug log.

6.3.3.11 TLE_RECV

TLE_RECV	
Message Description	This message indicates that the two line element MSD from the file indicated above was processed successfully and any downstream processing using the file can commence.
Content	TLE MSD product *file name* received
Variable Data	The file name of the product as indicated above
Recommended Action	No action needed

6.3.3.12 TLE_FAIL

TLE_FAIL	
Message Description	This message indicates that the two line element MSD from the file indicated above was not processed due to some sort of error condition.
Content	TLE MSD product *file name* received
Variable Data	The file name of the product as indicated above
Recommended Action	Check for common error conditions according to the relevant standard operating procedure. Depending on the repeatability of the error, the debug log could be enabled to provide a detailed debug log.

7.0 APPENDIX A - SOFTWARE MAINTENANCE

The following section defines development specific information on how to modify the ING code for specialized topics.

7.1 UPDATING PROCESSING COEFFICIENTS

The mission support data associated with processing coefficients is frequently altered and as such the code handling these coefficients must be changed often as well. These updates are driven by changes to PRO algorithms. As DQM is the provider of these products, they must be informed of the changes as well. The following section outlines the ways in which the operational ING software and unit tests must be changed to accommodate new coefficients. Prior to enacting any changes, care must be taken to coordinate with the PRO SI to ensure that any ING changes are merged at the appropriate time. Since PRO compiles using the ING defined structure, if there is a mismatch between the ING and PRO baselines, it will generally cause the build to fail.

7.1.1 Changes to values only

Changes to individual values in the Processing Coefficient file do not require any changes in the ING software. Value changes are made by the DQM subsystem and provided to the ING landing zone to update DMS with a new version for PRO to use per normal operations..

7.1.2 Adding/Deleting parameters

7.1.2.1 Updating the structure file

The first thing to update is the basic structure definition that represents the internal binary format of the. Generally this is done when the structure is provided as part of an algorithm drop or tech memo direction and is provided to the ING SI by the PRO SI. If it is not, the changes must be manually added to the coefficient structure as found in /vobs/ING/include. For instance, say there is a simple structure with the form:

```
struct IngMsdCoefficients_SampleStruct
{
   int a_value;
};
```

If a tech memo added the use of a new floating point coefficient named b_value then the structure would be modified to the following:

```
struct IngMsdCoefficients_SampleStruct
```

```
{
   int a_value;
   float b_value;
};
```

7.1.2.2 Updating the converter

The next step is to update the converter. This is the code responsible for translating a value from the native XML format to our internal binary format. The manner in which the code is modified depends on whether or not the new coefficient values are arrays or stand-alone values. For all Processing Coefficient files, there will be a converter named as:

/vobs/ING/MSD/Coefficients/src/IngMsdCoefficients_<AlgName>Converter.cpp

The conversion process is handled by a virtual function that must be defined in the converter called *assembleEntry().* This method requires specific calls to convert each XML tag into the binary structure. Currently there is generic code to handle the conversion of stand-alone values and dimensional arrays with up to three dimensions.

7.1.2.2.1 Stand-alone values

For each stand alone value in the XML, *assembleEntry()* needs the following line inserted:

```
getCoefValueStream("a_value")
  >> currentStruct->a_value;
```

7.1.2.2.2 Arrays

7.1.2.2.2.1 One dimensional

For arrays with only one dimension, the following needs to be added to *assembleEntry():*

```
rowList = getCoefNameList("a_value", A_VALUE_ARRAY_SIZE);
for(int iRow = 0;iRow < A_VALUE_ARRAY_SIZE;iRow++)
{
   getCoefValueStream(rowList->at(iRow))
      >> currentStruct->a_value[iRow];
}
delete rowList;
rowList = 0;
```

A_VALUE_ARRAY_SIZE is a placeholder for the size of the array being added. Sometimes these array sizes are explicit (i.e. ‘4’) and sometimes they are defined by a const or enum (i.e. NUM_SCANS). The code can use either method.
NOTE: If this is the first one dimensional array within the structure, rowList will need to be defined before the conversion as the following”

```
std::vector<string>* rowList = 0;
```

7.1.2.2.2.2 Two dimensional

For arrays with two dimensions, the following needs to be added to *assembleEntry():*

```
 rowList = getCoefNameList("a_value", A_VALUE_ROW_SIZE);
for (int iRow = 0; iRow < A_VALUE_ROW_SIZE; iRow++)
{
   colList = getCoefNameList(rowList->at(iRow), A_VALUE_COL_SIZE);
   for (int iCol = 0; iCol < A_VALUE_COL_SIZE; iCol++)
   {
      getCoefValueStream(colList->at(iCol))
         >> currentStruct-> a_value[iRow][iCol];
   }

   delete colList;
   colList = 0;
}
delete rowList;
rowList = 0;
```

A_VALUE_ROW_SIZE is a placeholder value for the dimension of the row portion of the two dimensional array being added and A_VALUE_COL_SIZE represents the dimension of the columns portion. Sometimes these sizes are explicit (IE ‘4’) and sometimes they are defined by a const or enum (IE NUM_SCANS). The code can interpret either way. If this is the first two dimensional array within the structure, colList and possibly rowList, will need to be defined as well with;

```
std::vector<string>* rowList = 0;
std::vector<string>* colList = 0;
```

7.1.2.2.2.3 Three Dimensional

Finally, for the rare three dimensional array, the following needs added to *assembleEntry():*

```
 rowList = getCoefNameList("a_value", A_VALUE_ROW_SIZE);
for (int iRow = 0; iRow < A_VALUE_ROW_SIZE; iRow++)
{
   colList = getCoefNameList(rowList->at(iRow), A_VALUE_COL_SIZE);
   for (int iCol = 0; iCol < A_VALUE_COL_SIZE; iCol++)
   {
      depthList = getCoefNameList(colList->at(iCol), A_VALUE_DEPTH_SIZE);
      for (int iDepth = 0;iDepth < NRAZ;iDepth++)
      {
         getCoefValueStream(depthList->at(iDepth))
            >> currentStruct-> a_value[iRow][iCol][iDepth];
      }

      delete depthList;
      depthList = 0;
   }
   delete colList;
   colList = 0;
}
delete rowList;
rowList = 0;
```

A_VALUE_ROW_SIZE is a placeholder for the dimension of the row portion of the two dimensional array being added, A_VALUE_COL_SIZE represents the dimension of the columns portion and A_VALUE_DEPTH_SIZE corresponds to the size of the third dimension. Sometimes these sizes are explicit (IE ‘4’) and sometimes they are defined by a const or enum (IE NUM_SCANS). The code can interpret either way. If this is the first three dimensional array within the structure, colList, rowList, and depthList will need to be defined as well with;

```
std::vector<string>* rowList = 0;
std::vector<string>* colList = 0;
std::vector<string>* depthList = 0;
```

7.1.3 New Processing Coefficient file

A new Processing Coefficient could occur due to a new algorithm being developed or simply an algorithm that didn’t have configurable parameters before. This adds some complexity to the update process shown above, but in general it is the same. In addition to creating a new structure and converter, the developer must update the *IngMsdCommon_ConversionFactory* class to include the new converter and add entries to the *IngMsdCommon_ShortNameCollection* for the new collection shortname of the processing coefficient. These items tie the new product into the ING MSD baseline for conversion.

7.1.4 Testing

7.1.4.1 Updating the unit test

Finally the unit test and extractor code must be updated to ensure that the changes made in the previous two sections were successful.

Each converter has a corresponding unit test located at:

/vobs/ING/MSD/Coefficients/unittest/<algorithm name>

Within the *ConverterUT.cpp* file, there is a *doDisplayStruct()*. Within the *doDisplayStruct()* method are a series of couts that print the contents of a converted coefficients file to the screen so they can be reviewed. The content of this code is extremely similar to the conversion code in the previous section. For example, to output an array the *doDisplayStruct(t* method uses the following code;

```
 for (int i1 = 0; i1 < NSZA; i1++)
{
   cout << "  [";
   for(int i2 = 0; i2 < NVZA; ++i2)
   {
      cout << " " << coefficientStruct_.A_win_over[i1][i2];
   }
   cout <<  "]" << endl;
}
```

Review of existing Processing Coefficients should explain the process fairly well. The largest problem area is checking to see how unsigned char's should be output. Sometimes they must be cast as unsigned ints in order to display as a number. Even more rarely, occasionally there are unsigned char arrays that function as primitive strings within a coefficients file. These require special case behavior as well.

After the ConverterUT.cpp file has been updated, the <algorithm name>Extractor.cpp file must be updated as well. This is the code responsible for reading a coefficient entry from DMS and displaying it to the screen in a format that is usable by DQM. Its format is extremely similar to that of the ConverterUT.cpp file and so the changes made to it will be virtually identical. One new cout statement will be needed for each coefficient entry being added.

One difference between the ConverterUT.cpp and the Extractor.cpp file is that the extraction code is sensitive to precision. A call to "cout.precision(8);" must be made prior to the printing of any float data and a call to " cout.precision(16);" must be made prior to the printing of any double data to ensure that precision is properly maintained.

Finally, the TestFile XML file within the unit test folder must be modified to add any new entries to the test XML. The XML format for coefficients is fairly simple. Each coefficient entry and each element within an array has an XML block that looks like the following;

```
<coeff description="" >
   <id>1</id>
   <name>sample_entry</name>
   <type>float</type>
   <value>0.1</value>
   <min>1</min>
   <max>100</max>
</coeff>
```

To add new coefficients, the XML must simply be expanded with each new element. Arrays are handled by using C++ 0-indexed notation. So the 'name' field for the first element in the two-dimensional array "sample_array" would contain sample_array[0][0]. For the value field, normally the values assigned are monotonically increasing for each coefficient.

7.1.4.2 Executing the unit test

Once the code and unit test files have been modified, the unit test can actually be run to verify that the changes were successful. The first step then is to compile the basic ING llibraries. To do so, run the script located at /vobs/ING/script/buildVobDebug.csh.
If the script has completed without any compile errors, the next script at /vobs/ING/script/buildUnittests.csh must be run. If that script completed successfully as well, the actual unit test may be built. To do so, change directories to the unittest directory located at /vobs/ING/MSD/Coefficients/unittest/<algorithm name>Converter/. Once there, type make program. This should compile the two unit test executables.

From here, the unit test can actually be run. To do so type 'make unittest'. Assuming that there were no build errors, the unit test is now ready to be run. To run the unittest, use the command sequence './ConverterTest –lw ../tk_config.xml'. The unit test should run successfully, in the process printing out a complete list of all the coefficients along with their values. This output should then be manually checked for correctness in light of any changes made to the structure.

7.1.5 Providing values to DQM

Since the ING MSD unit test is capable of interpreting binary coefficient files, ING usually provides DQM any information it needs when coefficient values change. The process for obtaining these values is fairly straightforward. Normally the PRO SI is provided or creates a binary coefficients file as part of any coefficient changes. The first step in providing coefficient values to DQM then is to import this binary coefficient file into DMS with the metadata N_Collection_Short_Name set to TestFile-Int to match the format expected by the unit test code.

Once that is done, the ExtractorTest program must be run with the following command 'ExtractorTest –lw ../tk_config.xml'. Once this is done, a series of coefficient names and values will be displayed in a specific format. This format is an informal format agreed to by both the ING SI and the DQM SI and is used as an input into a DQM script. Therefore, the values displayed on screen must be copied and provided to whoever the DQM point of contact is for these coefficient changes.